



QR Code Tutorial

How to generate QR code,
IDFS lecture supplement

Author: Rozhdestvenskiy Dmitry, revision: Petr Bureš
Based on <http://www.thonky.com/qr-code-tutorial/>

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

Version	Modules	ECC Level	Data bits	Numeric	Alphanumeric	Binary	Kanji
1	21x21	L	152	41	25	17	10
		M	128	34	20	14	8
		Q	104	27	16	11	7
		H	72	17	10	7	-
.....							
40	177x177	L	23,648	7,089	4,296	2,953	1,817
		M	18,672	5,596	3,391	2,331	1,435
		Q	13,328	3,993	2,420	1,663	1,024
		H	10,208	3,057	1,852	1,273	784

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**

Bit string	Data mode
0001	Numeric Mode
0010	Alphanumeric Mode
0100	Binary Mode
1000	Japanese Mode

Version	1-9	10-26	27-40
Numeric (bits)	10	12	14
Alphanumeric (bits)	9	11	13
Binary (bits)	8	16	16
Japanese (bits)	8	10	12

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.

0 0	A 10	K 20	U 30	+ 40
1 1	B 11	L 21	V 31	- 41
2 2	C 12	M 22	W 32	. 42
3 3	D 13	N 23	X 33	/ 43
4 4	E 14	O 24	Y 34	: 44
5 5	F 15	P 25	Z 35	
6 6	G 16	Q 26	(space) 36	
7 7	H 17	R 27	\$ 37	
8 8	I 18	S 28	% 38	
9 9	J 19	T 29	* 39	

	HE	LL	O(space)	WO	RL	D
	$(45*17)+14$	$(45*21)+21$	$(45*24)+36$	$(45*32)+24$	$(45*27)+21$	13
	779	966	1116	1464	1236	13
0010 000001011	01100001011	01111000110	10001011100	10110111000	10011010100	001101

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.
 - *Our string is 74 bits long, we add four 0s to the end (for 102 add just 2).*
 - Our binary string so far: 0010 000001011 01100001011 01111000110 10001011100 10110111000 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 01111000 11010001 01110010 11011100 01001101 01000011 01000000*

terminator

Generate a binary string 4

- 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 00001011 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, 236U
 - 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:
 - $(\text{number of data blocks}) + (\text{number of error correction code words}) - 1$
 - *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 $(x - \alpha)(x - \alpha^2) \dots (x - \alpha^t)$, 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.*
 - More info at: <http://www.thonky.com/qr-code-tutorial/part-2-error-correction/>
 - Result of these mathematical operation : $168x^{12} + 72x^{11} + 22x^{10} + 82x^9 + 217x^8 + 54x^7 + 156x^6 + 0x^5 + 46x^4 + 15x^3 + 180x^2 + 122x^1 + 16x^0 \dots$ **Next we convert it to alpha notation**
 - Computed correction coefficients 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.
 - Nice calculator <http://www.pclviewer.com/rs2/calculator.html>

QR Mask Patterns Explained

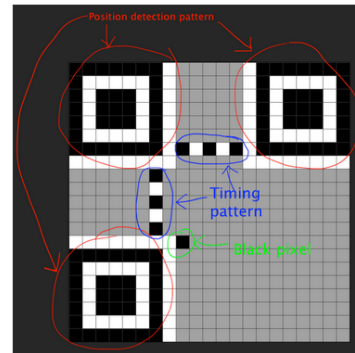
- There are eight mask patterns that can be used to change the outputted matrix.
- Each mask pattern changes the bits according to their coordinates in the QR matrix.
- The purpose of a mask pattern is to make the QR code **easier for a QR scanner to read**.
- Each mask pattern uses a formula to determine 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.

Mask Number	When to switch the bit
0	$(y + x) \bmod 2 == 0$
1	$y \bmod 2 == 0$
2	$x \bmod 3 == 0$
3	$(y + x) \bmod 3 == 0$
4	$((y / 2) + (x / 3)) \bmod 2 == 0$
5	$((y * x) \bmod 2) + ((y * x) \bmod 3) == 0$
6	$((y * x) \bmod 2) + ((y * x) \bmod 3) \bmod 2 == 0$
7	$((y + x) \bmod 2) + ((y * x) \bmod 3) \bmod 2 == 0$

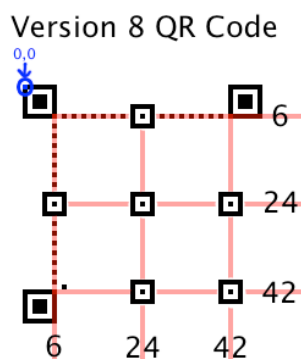
Generate the QR Code

- 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 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.



QR Version 2	6	18				
QR Version 3	6	22				
QR Version 4	6	26				
QR Version 5	6	30				
QR Version 6	6	34				
QR Version 7	6	22	38			
QR Version 8	6	24	42			
QR Version 9	6	26	46			
QR Version 10	6	28	50			
QR Version 11	6	30	54			
QR Version 12	6	32	58			
QR Version 13	6	34	62			
QR Version 14	6	26	46	66		
QR Version 15	6	26	48	70		
QR Version 16	6	26	50	74		
QR Version 17	6	30	54	78		

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.

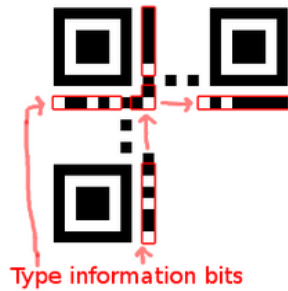


Table displays the type information bits that are required for the different error correction levels and mask patterns.

ECC Level	Mask Pattern	Type Information Bits
L	0	111011111000100
L	1	111001011110011
L	2	111110110101010
L	3	111100010011101
L	4	110011000101111
L	5	110001100011000
L	6	110110001000001
L	7	110100101110110
M	0	101010000010010
M	1	101000100100101
M	2	101111001111100
M	3	101101101001011
M	4	100010111111001
M	5	100000011001110
M	6	100111110010111
M	7	100101010100000
Q	0	011010101011111
Q	1	011000001101000

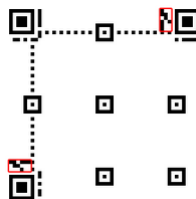
Add Version Information

- 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.

00	03	06	09	12	15
01	04	07	10	13	16
02	05	08	11	14	17

00	01	02
03	04	05
06	07	08
09	10	11
12	13	14
15	16	17

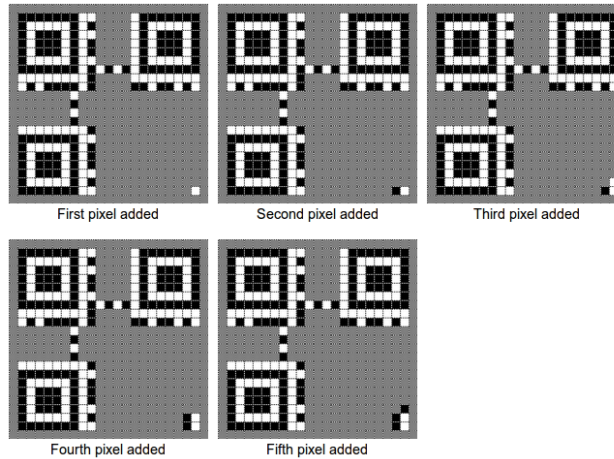


Version	Bits for Version Information
7	001010010011111000
8	000111101101000100
9	100110010101100100
10	011001011001010100
11	01101111101110100
12	001000110111001100
13	111000100001101100
14	010110000011011100
15	000101001001111100
16	000111101101000010
17	010111010001100010
18	111010000101010010
19	001001100101110010
20	011001011001001010
21	011000001011101010
22	100100110001011010
23	000110111111111010
24	001000110111000110
25	000100001111100110
26	110101011111010110

Add Data Bits

Upward Column

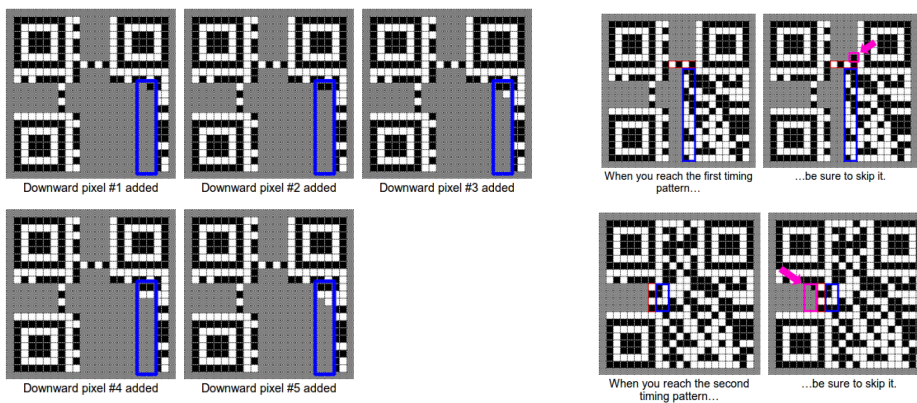
- The data bits are added in a particular order. two-pixel column continues to go upward, skipping any pixels that are already set, until it reaches the top of the QR code.



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,

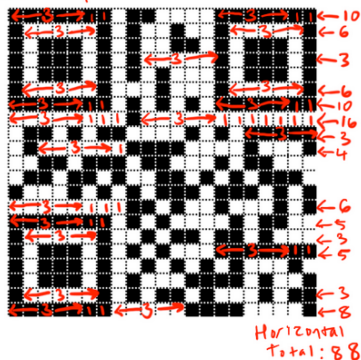
There are four penalty rules:

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.**

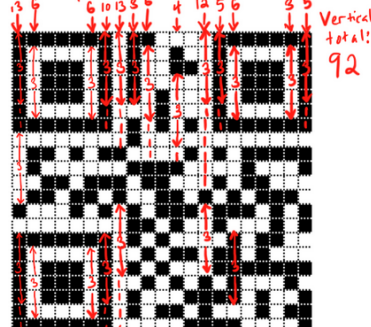
Penalty Rule 1

- 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.

Five consecutive squares get a penalty of 3.
Each consecutive square after that adds 1 to the penalty.

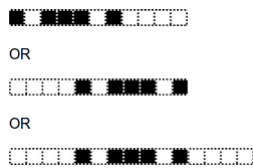


Five consecutive squares get a penalty of 3.
Each consecutive square after that adds 1 to the penalty.

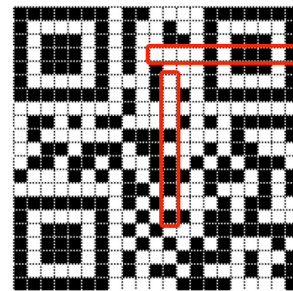
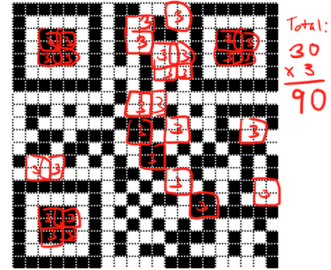


Penalty Rule 2 AND 3

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

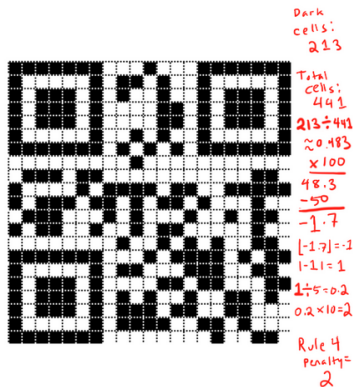


Each time this pattern is found, you add 40 to the penalty score.

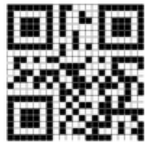
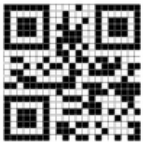
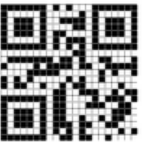
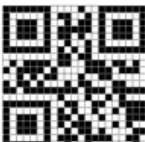
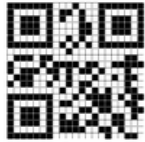

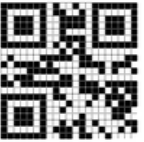



Penalty Rule 4

- 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.



Choose the best one

 Mask Pattern 0 Penalty 1: 180 Penalty 2: 90 Penalty 3: 80 Penalty 4: 0 Total: 350	 Mask Pattern 1 Penalty 1: 172 Penalty 2: 129 Penalty 3: 120 Penalty 4: 0 Total: 421	 Mask Pattern 2 Penalty 1: 206 Penalty 2: 141 Penalty 3: 160 Penalty 4: 0 Total: 507	 Mask Pattern 6 Penalty 1: 171 Penalty 2: 102 Penalty 3: 80 Penalty 4: 4 Total: 357
 Mask Pattern 3 Penalty 1: 180 Penalty 2: 141 Penalty 3: 120 Penalty 4: 2 Total: 443	 Mask Pattern 4 Penalty 1: 195 Penalty 2: 138 Penalty 3: 200 Penalty 4: 0 Total: 553	 Mask Pattern 5 Penalty 1: 189 Penalty 2: 156 Penalty 3: 200 Penalty 4: 2 Total: 547	 Mask Pattern 7 Penalty 1: 197 Penalty 2: 123 Penalty 3: 200 Penalty 4: 0 Total: 520

Thank you for your attention

- Reference: This presentation based on materials from the site <http://www.thonky.com/qr-code-tutorial/>
- Reed Solomon code calculator <http://www.pclviewer.com/rs2/calculator.html>

Voluntary Homework assignment:

- Choose short phrase, (up to 16 characters)
- “hand” encode it into QR code using the tutorial webpages.
- Present in a form of written report.