**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ SÀI GÒN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**\_\_\_oOo\_\_\_**



**BÁO CÁO ĐỀ TÀI**

**NGHIÊN CỨU VÀ TRIỂN KHAI MÃ QR CÓ BẢO MẬT BẰNG THUẬT TOÁN MÃ HÓA BẤT ĐỐI XỨNG**

Sinh viên thực hiện:

|  |  |  |
| --- | --- | --- |
| MSSV | Họ và Tên | Lớp |
| DH52201426 | Lê Hoàng Thân | D22\_TH02 |

**THÀNH PHỐ HỒ CHÍ MINH – 2025**

1. **Tóm tắt**
   1. Tóm tắt ngắn gọn mục đích, phạm vi và kết quả nghiên cứu.
2. **Giới thiệu**
   1. Lý do chọn đề tài
   2. Mục tiêu nghiên cứu
   3. Phạm vi và giới hạn nghiên cứu
   4. Tổng quan về mã QR và ứng dụng
3. **Mã QR (QR Code)**
   1. Khái niệm, lịch sử phát triển
   2. Cấu trúc và nguyên lý hoạt động
   3. Các chuẩn mã hóa và lưu trữ thông tin
4. **Thuật toán mã hóa**
   1. Giới thiệu thuật toán bạn sử dụng (ví dụ: RSA, AES, ECC, …)
   2. Nguyên lý hoạt động
   3. Tính năng bảo mật và ứng dụng
5. **Phân tích yêu cầu và thiết kế hệ thống**
   1. Yêu cầu chức năng
   2. Yêu cầu phi chức năng (bảo mật, hiệu năng…)
   3. Kiến trúc tổng quan hệ thống
   4. Thiết kế chi tiết các thành phần liên quan đến mã QR và mã hóa
6. **Triển khai**
   1. Môi trường phát triển
   2. Mô tả chi tiết cách tạo mã QR trong dự án
   3. Mô tả cách thuật toán mã hóa được tích hợp và vận hành
   4. Các bước và ví dụ minh họa
7. **Kiểm thử và đánh giá**
   1. Phương pháp kiểm thử
   2. Kết quả kiểm thử
   3. Đánh giá ưu nhược điểm
8. **Kết luận và hướng phát triển**
   1. Tổng kết những gì đã làm được
   2. Hướng mở rộng và cải tiến trong tương lai
9. **Tài liệu tham khảo**
10. **Phụ lục** (nếu có)
    1. Code mẫu
    2. Bảng dữ liệu
    3. Hình ảnh minh họa

**MỤC LỤC**

# **Mã phản hồi nhanh QR Code**

## **Tổng quan về QR Code**

Định dạng mã QR được tạo ra vào năm 1994 bởi công ty Denso-Wave của Nhật Bản , một công ty con của Toyota chuyên sản xuất linh kiện ô tô. Tiêu chuẩn được xác định trong ISO/IEC 18004:2006 . Việc sử dụng mã QR là không có giấy phép.

Mã QR nhỏ nhất là 21x21 pixel và lớn nhất là 177x177. Các kích thước được gọi là phiên bản “version”. Kích thước 21x21 pixel là version 1, 25x25 là version 2, ... Kích thước 177x177 là version 40.

Ngoài ra, mã QR còn bao gồm tính năng sửa lỗi: khi bạn mã hóa mã QR, bạn cũng tạo ra một số dữ liệu dư thừa giúp trình đọc QR đọc mã chính xác ngay cả khi một phần của mã không thể đọc được. Có bốn cấp độ sửa lỗi mà bạn có thể chọn. Mức thấp nhất là L, cho phép đọc mã ngay cả khi 7% trong số đó không thể đọc được. Sau đó là M, cung cấp khả năng sửa lỗi 15%, sau đó là Q, cung cấp 25% và cuối cùng là H, cung cấp 30%.

Dung lượng của một mã QR nhất định tùy thuộc vào phiên bản và mức độ sửa lỗi cũng như loại dữ liệu bạn đang mã hóa. Có bốn chế độ dữ liệu mà mã QR có thể mã hóa: số, chữ và số, nhị phân hoặc Kanji. Danh sách các phiên bản QR của trang web Denso-Wave bao gồm thông tin về số lượng bit dữ liệu bạn có thể mã hóa trong mỗi phiên bản.

## **Tổng quan về cách tạo mã QR**

**Bước 1: Phân tích dữ liệu**

Mã QR mã hóa một chuỗi văn bản. Chuẩn QR có 4 chế độ mã hóa văn bản: số, chữ và số, byte và Kanji. Mỗi chế độ mã hóa văn bản dưới dạng một chuỗi bit (1 và 0), nhưng mỗi chế độ sử dụng một phương pháp khác nhau để chuyển đổi văn bản thành bit và mỗi phương pháp mã hóa được tối ưu hóa để mã hóa dữ liệu với chuỗi bit ngắn nhất có thể. Do đó, bước đầu tiên là thực hiện **phân tích dữ liệu** để xác định xem văn bản có thể được mã hóa ở chế độ số, chữ và số, byte hay Kanji hay không, sau đó chọn chế độ tối ưu nhất cho văn bản.

**Bước 2: Mã hóa dữ liệu.**

Sau khi đã chọn chế độ mã hóa thích hợp cho văn bản, bước tiếp theo là mã hóa văn bản. Phần **mã hóa dữ liệu** mô tả chi tiết quá trình này cho từng chế độ mã hóa. Kết quả của bước này là một chuỗi bit được chia thành các từ mã dữ liệu “data codeword”, mỗi codeword có độ dài 8 bit.

**Bước 3: Mã hóa sửa lỗi.**

Mã QR sử dụng tính năng sửa lỗi. Điều này có nghĩa là sau khi tạo chuỗi bit dữ liệu đại diện cho văn bản, phải sử dụng các bit đó để tạo ra các từ mã dùng để sửa lỗi “error correction codeword” bằng quy trình sửa lỗi “Reed-Solomon error correction”.

Trình quét mã QR đọc cả “data codeword” và “error correction codeword”. Bằng cách so sánh cả hai, trình quét mã có thể xác định xem nó có đọc dữ liệu chính xác hay không và có thể sửa lỗi nếu không đọc dữ liệu chính xác. Phần **mã hóa sửa lỗi** sẽ giải thích chi tiết quá trình tạo từ mã sửa lỗi.

**Bước 4: Cấu trúc Final Message.**

“Data codeword” và “error correction codeword” được tạo ở các bước trước bây giờ phải được sắp xếp theo đúng thứ tự. Đối với các mã QR lớn, “data codeword” và “error correction codeword” được tạo thành các khối và các khối này phải được xen kẽ theo đặc tả mã QR. Quá trình này sẽ được giải thích trong phần **cấu trúc Final Message**.

**Bước 5: Vị trị module trong ma trận.**

Sau khi tạo “data codeword” và “error correction codeword” và sắp xếp chúng theo đúng thứ tự, cần phải đặt các bit vào ma trận mã QR. Các codeword được sắp xếp trong ma trận một cách cụ thể. Trong bước này, cũng sẽ đặt các mẫu “pattern” chung cho tất cả các mã QR, chẳng hạn như các “box” ở ba góc. Quá trình này sẽ được giải thích chi tiết trong phần **sắp xếp module trong phần ma trận**.

**Bước 6: Che giấu dữ liệu.**

Một số “pattern” nhất định trong ma trận mã QR có thể gây khó khăn cho trình quét mã QR đọc mã chính xác. Để khắc phục điều này, đặc tả mã QR xác định tám mẫu mặt nạ “mask pattern”, mỗi “pattern” thay đổi mã QR theo một mẫu cụ thể. Cần phải xác định mẫu “mask pattern” nào dẫn đến mã QR có ít đặc điểm không mong muốn nhất. Điều này được thực hiện bằng cách đánh giá từng ma trận bị che giấu dựa trên bốn quy tắc phạt “penalty rule”. Mã QR cuối cùng của phải sử dụng“mask pattern” dẫn đến điểm phạt thấp nhất. Quá trình che giấu sẽ được giải thích trong phần **che giấu dữ liệu**.

**Bước 7: Thông tin định dạng và phiên bản.**

Bước cuối cùng là thêm thông tin “format” và “version” (nếu cần) vào mã QR bằng cách thêm pixel vào các vùng cụ thể của mã bị bỏ trống ở các bước trước. Các “format pixel” xác định mức độ sửa lỗi và “mask pattern” đang được sử dụng trong mã QR này. Các “version pixel” mã hóa kích thước của ma trận QR và chỉ được sử dụng trong các mã QR lớn hơn. Để biết chi tiết về bước cuối cùng này được giải thích trong phần **thông tin về định dạng và phiên bản**.

## **Phân tích dữ liệu**

* + 1. **Các chế độ mã QR**

Bốn chế độ mã hóa bao gồm các ký tự sau:

**1. Chế độ số “Numeric mode”** dành cho các chữ số thập phân từ 0 đến 9.

**2. Chế độ chữ và số “Alphanumeric mode”** dành cho các chữ số thập phân từ 0 đến 9, cũng như các chữ cái viết hoa (không phải chữ thường!) và các ký hiệu $, %, ×, +, -, ., /, và : cũng như khoảng trắng. Tất cả các ký tự được hỗ trợ cho chế độ chữ và số đều được liệt kê ở cột bên trái của bảng chữ và số này .

**3. Chế độ byte “Byte mode”**, theo mặc định, dành cho các ký tự từ bộ ký tự ISO-8859-1. Tuy nhiên, một số máy quét mã QR có thể tự động phát hiện xem UTF-8 có được sử dụng ở chế độ byte hay không.

**4. Chế độ Kanji** **“Kanji mode”** dành cho các ký tự double-byte từ bộ ký tự Shift JIS. Mặc dù UTF-8 có thể mã hóa các ký tự Kanji nhưng nó phải sử dụng ba hoặc bốn byte để làm như vậy. Mặt khác, Shift JIS chỉ sử dụng hai byte để mã hóa mỗi ký tự Kanji, do đó chế độ Kanji nén các ký tự Kanji hiệu quả hơn. Nếu toàn bộ chuỗi đầu vào bao gồm các ký tự trong phạm vi byte kép của Shift JIS, hãy sử dụng chế độ Kanji. Cũng có thể sử dụng nhiều chế độ trong cùng một mã QR.

**Chế độ Phiên dịch kênh mở rộng (ECI)** chỉ định trực tiếp bộ ký tự (ví dụ: UTF-8). Tuy nhiên, một số trình đọc mã QR không hỗ trợ chế độ ECI và sẽ không hiểu mã QR sử dụng nó.

**Chế độ Nối** **có cấu trúc** **“Structured Append mode”** mã hóa dữ liệu trên nhiều mã QR, tối đa 16 mã QR.

**Chế độ FNC1** cho phép mã QR hoạt động như mã vạch GS1.

* + 1. **Lưu ý về chế độ Kanji**

Một số trình đọc mã QR có thể nhận ra khi UTF-8 được sử dụng ở chế độ byte. Vì tất cả các ký tự Shift JIS đều được biểu thị bằng UTF-8 nên có thể sử dụng chế độ byte cho Kanji với mã hóa UTF-8.

Tuy nhiên, Kanji trong UTF-8 được mã hóa bằng ba byte (hoặc bốn, trong một số trường hợp hiếm hoi), trong khi các ký tự Shift JIS được mã hóa bằng hai hoặc một byte. Nói cách khác, sẽ không thể ghép nhiều ký tự vào mã QR nếu sử dụng UTF-8 ở chế độ byte cho Kanji. Sử dụng chế độ Kanji cho Shift JIS Kanji cho công suất cao nhất.

Vì vậy, việc sử dụng chế độ Kanji cho Kanji hay không là tùy thuộc vào nhu cầu của người dùng.

* + 1. **Lưu ý về UTF-8**

Một số trình đọc mã QR tự động phát hiện nếu UTF-8 được sử dụng ở chế độ byte, nhưng những trình đọc không có chức năng này có thể hiển thị các ký tự không chính xác nếu sử dụng UTF-8 ở chế độ byte. Để khắc phục điều này, có thể sử dụng chế độ ECI, như đã đề cập ở trên, cho phép chỉ định một bộ ký tự khác với bộ ký tự ISO-8859-1 mặc định ở chế độ byte. Không phải tất cả các trình đọc mã QR đều hỗ trợ chế độ ECI.

Một tùy chọn khác là đặt dấu thứ tự byte UTF-8 (BOM) trước văn bản đầu vào. Một số trình đọc mã QR sẽ đọc dấu thứ tự byte và hiểu rằng văn bản được mã hóa bằng UTF-8. Không phải tất cả các trình đọc mã QR đều có thể đọc chính xác. Dấu thứ tự byte cho UTF-8 là một tập hợp gồm ba số, được hiển thị ở đây dưới dạng thập lục phân: 0xEF 0xBB 0xBF

* + 1. **Cách chọn chế độ hiệu quả nhất**

Để chọn chế độ hiệu quả nhất cho mã QR, cần kiểm tra các ký tự trong chuỗi đầu vào và kiểm tra các điều kiện sau.

Nếu chuỗi đầu vào chỉ bao gồm các chữ số thập phân (0 đến 9),cần sử dụng **“numeric mode”.**

Nếu “numeric mode” không được áp dụng và tất cả các ký tự trong chuỗi đầu vào có thể được tìm thấy ở cột bên trái của bảng chữ và số, cần sử dụng **“alphanumeric mode”**. Chữ thường KHÔNG THỂ được mã hóa ở chế độ chữ và số; chỉ viết hoa.

Nếu có một ký tự không nằm ở cột bên trái của bảng chữ và số nhưng có thể được mã hóa theo ISO 8859-1, hãy sử dụng **“byte mode”**. Như đã đề cập ở trên, trình đọc mã QR có thể nhận dạng UTF-8 ở chế độ byte.

Nếu tất cả các ký tự đều nằm trong bộ ký tự Shift JIS thì sử dụng **“Kanji mode”**. Thay vào đó, các ký tự Shift JIS có thể được mã hóa bằng UTF-8, do đó có thể sử dụng **“byte mode”** cho Kanji, nhưng nhìn chung sẽ hiệu quả hơn khi sử dụng Shift JIS và sử dụng “Kanji mode” cho các ký tự Kanji.

* + 1. **Chế độ trộn và tối ưu hóa**

Có thể sử dụng nhiều chế độ trong một mã QR bằng cách thêm chỉ báo chế độ “mode indicator” trước mỗi phần byte sử dụng chế độ đó. Đặc tả mã QR giải thích cách chuyển đổi chế độ một cách tối ưu nhất.

* + 1. **Kết luận**

Bằng cách kiểm tra các ký tự trong văn bản đầu vào, có thể chọn chế độ tối ưu nhất để mã hóa văn bản đó. Cần xem xét các hạn chế của trình đọc mã QR khi chọn chế độ và không phải tất cả các trình đọc mã QR đều tuân thủ tiêu chuẩn. Hơn nữa, cần xem xét nhu cầu của người dùng khi quyết định nên sử dụng chế độ Kanji hay sử dụng UTF-8 ở chế độ byte thay thế.

## **Mã hóa dữ liệu**

Mỗi chế độ mã hóa được thiết kế để tạo chuỗi bit ngắn nhất có thể cho các ký tự được sử dụng trong chế độ đó. Mỗi chế độ sử dụng một phương pháp khác nhau để chuyển đổi văn bản đầu vào thành một chuỗi bit.

**Bước 1: Chọn mức sửa lỗi**

Trước khi mã hóa dữ liệu, cần chọn mức sửa lỗi. Như đã đề cập ở phần giới thiệu, mã QR sử dụng tính năng sửa lỗi “Reed-Solomon error correction”. Quá trình này tạo ra các “error correction codeword” (byte) dựa trên dữ liệu được mã hóa. Trình đọc mã QR có thể sử dụng các byte sửa lỗi này để xác định xem nó có đọc dữ liệu chính xác hay không và “error correction codeword” có thể được sử dụng để sửa các lỗi đó. Có bốn cấp độ sửa lỗi: L, M, Q, H.

|  |  |
| --- | --- |
| **Mức độ sửa lỗi** | **Khả năng sửa lỗi** |
| L | Phục hồi 7% dữ liệu |
| M | Phục hồi 15% dữ liệu |
| Q | Phục hồi 25% dữ liệu |
| H | Phục hồi 30% dữ liệu |

Lưu ý: mức sửa lỗi càng cao thì yêu cầu nhiều byte hơn, do đó, mức sửa lỗi càng cao thì mã QR sẽ càng lớn.

**Bước 2: Xác định “version” nhỏ nhất cho dữ liệu**

Các kích thước khác nhau của mã QR được gọi là phiên bản “version”. Có bốn mươi “version” có sẵn. “Version” nhỏ nhất là “version 1” và có kích thước 21 × 21 pixel. “Version 2” có kích thước 25 × 25 pixel. Phiên bản lớn nhất là “version 40” và có kích thước 177 × 177 pixel. Mỗi “version” lớn hơn 4 pixel so với “version” trước.

Mỗi “version” đều có dung lượng tối đa, tùy theo chế độ sử dụng. Ngoài ra, mức độ sửa lỗi còn hạn chế dung lượng hơn nữa. “Bảng dung lượng ký tự” liệt kê dung lượng của tất cả các phiên bản QR cho một chế độ mã hóa và mức sửa lỗi nhất định.

**Cách xác định phiên bản nhỏ nhất:**

Tại thời điểm này, đếm số ký tự cần mã hóa và xác định phiên bản nào nhỏ nhất có thể chứa số ký tự đó cho chế độ mã hóa và mức sửa lỗi mong muốn.

Ví dụ: cụm từ HELLO WORLD có 11 ký tự. Nếu mã hóa nó bằng sửa lỗi cấp Q, bảng dung lượng ký tự cho biết mã phiên bản 1 sử dụng sửa lỗi cấp Q có thể chứa 16 ký tự ở chế độ chữ và số, vì vậy phiên bản 1 là phiên bản nhỏ nhất có thể chứa số ký tự này. Nếu cụm từ dài hơn 16 ký tự, chẳng hạn như HELLO THERE WORLD (có 17 ký tự), phiên bản 2 sẽ là phiên bản nhỏ nhất.

**Giới hạn:**

Mã QR dung lượng cao nhất là 40-L (phiên bản 40, mức sửa lỗi L). Dưới đây là bảng liệt kê dung lượng của mã QR 40-L cho bốn chế độ mã hóa. Đây là số ký tự tối đa có thể có mà một mã QR có thể chứa. Các phiên bản 40-M, 40-Q và 40-H có dung lượng thấp hơn vì chúng yêu cầu nhiều không gian hơn cho nhiều từ mã sửa lỗi hơn. Xem bảng dung lượng của tất cả các phiên bản ở **bảng dung lượng ký tự**.

|  |  |
| --- | --- |
| **Chế độ mã hóa** | **Số ký tự tối đa mà mã 40-L có thể chứa trong chế độ đó** |
| Numeric mode | 7089 ký tự |
| Alphanumeric mode | 4296 ký tự |
| Byte mode | 2953 ký tự |
| Kanji mode | 1817 ký tự |

**Bước 3: Thêm chỉ báo chế độ “Mode Indicator”**

Mỗi chế độ mã hóa có một chỉ báo chế độ “mode indicator” 4 bit để xác định nó. Dữ liệu được mã hóa phải bắt đầu bằng chỉ báo chế độ thích hợp để chỉ định chế độ đang được sử dụng cho các bit theo sau nó. Bảng sau liệt kê các chỉ báo chế độ cho từng chế độ.

Ví dụ: nếu mã hóa HELLO WORLD ở “Alphanumeric mode”, chỉ báo chế độ là 0010.

|  |  |
| --- | --- |
| **Tên chế độ** | **Chỉ báo chế độ** |
| Numeric mode | 0001 |
| Alphanumeric mode | 0010 |
| Byte mode | 0100 |
| Kanji mode | 1000 |
| ECI mode | 0111 |

**Bước 4: Thêm chỉ báo đếm kí tự “Character Count Indicator”**

Chỉ báo đếm ký tự “Character Count Indicator” là một chuỗi bit biểu thị số lượng ký tự đang được mã hóa. Chỉ báo đếm ký tự phải được đặt sau chỉ báo chế độ “Mode Indicator”. Hơn nữa, “Character Count Indicator” phải dài một số bit nhất định, tùy thuộc vào phiên bản QR.

Đếm số ký tự trong văn bản đầu vào ban đầu, sau đó chuyển số đó thành số nhị phân. Độ dài của “Character Count Indicator” tùy thuộc vào chế độ mã hóa và phiên bản mã QR sẽ được sử dụng. Để làm cho chuỗi nhị phân có độ dài thích hợp, đệm chuỗi đó vào bên trái bằng số 0.

Danh sách sau đây chứa kích thước của chỉ báo đếm ký tự cho từng chế độ và phiên bản.

Ví dụ: nếu mã hóa HELLO WORLD ở mã QR “version 1”và “alphanumeric mode” thì chỉ báo đếm ký tự phải dài 9 bit. Số ký tự của HELLO WORLD là 11. Trong hệ nhị phân, 11 là 1011. Đặt nó ở bên trái để tạo thành 9 bit dài:

000001011.

Đặt ký tự này sau chỉ báo chế độ từ bước 3 để có chuỗi bit sau:

0010 000001011.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Numeric mode** | **Alphanumeric mode** | **Byte mode** | **Kanji mode** |
| 1 - 9 | 10 bit | 9 bit | 8 bit | 8 bit |
| 10 - 26 | 12 bit | 11 bit | 16 bit | 10 bit |
| 27 - 40 | 14 bit | 13 bit | 16 bit | 12 bit |

**Bước 5: Mã hóa bằng chế độ đã chọn.**

Quá trình mã hóa cho mỗi chế độ mã hóa là khác nhau.

HELLO WORLD được mã hóa ở chế độ chữ và số “Alphanumeric Mode Encoding”.

Tiếp tục ví dụ HELLO WORLD.

|  |  |  |
| --- | --- | --- |
| **Alphanumeric Mode Encoding** | | |
| 0 | 0 | **Bước 1: Chia thành từng cặp**  Chia chuỗi thành các cặp ký tự:  HE, LL, O, WO, RL, D  **Bước 2: Tạo một số nhị phân cho mỗi cặp**  Đối với chế độ chữ số, mỗi ký tự chữ số được biểu diễn bằng một số. Cột bên trái hiển thị ký tự chữ số và cột bên phải hiển thị số biểu diễn ký tự đó.  Đối với mỗi cặp ký tự, lấy số biểu diễn (từ bảng chữ cái số) của ký tự đầu tiên và nhân nó với 45. Sau đó cộng số đó với số biểu diễn của ký tự thứ hai.  Ví dụ, cặp đầu tiên trong HELLO WORLD là HE:  H → 17  E → 14  Thực hiện theo các bước ở đoạn trước, nhân số đầu tiên với 45, sau đó cộng với số thứ hai:  (45 × 17) + 14 = 779  Bây giờ hãy chuyển đổi số đó thành chuỗi nhị phân 11 bit, thêm số 0 vào bên trái nếu cần.  779 → 01100001011  Nếu đang mã hóa một số lượng ký tự lẻ, như ví dụ này, thì lấy biểu diễn số của ký tự cuối cùng và chuyển đổi nó thành chuỗi nhị phân 6 bit. |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 5 | 5 |
| 6 | 6 |
| 7 | 7 |
| 8 | 8 |
| 9 | 9 |
| A | 10 |
| B | 11 |
| C | 12 |
| D | 13 |
| E | 14 |
| F | 15 |
| G | 16 |
| H | 17 |
| I | 18 |
| J | 19 |
| K | 20 |
| L | 21 |
| M | 22 |
| N | 23 |
| O | 24 |
| P | 25 |
| Q | 26 |
| R | 27 |
| S | 28 |
| T | 29 |
| U | 30 |
| V | 31 |
| W | 32 |
| X | 33 |
| Y | 34 |
| Z | 35 |
|  | 36 |
| $ | 37 |
| % | 38 |
| × | 39 |
| + | 40 |
| - | 41 |
| . | 41 |
| / | 42 |
| : | 44 |

Chuỗi bit cho đến nay là:

|  |  |  |
| --- | --- | --- |
| **Mode Indicator** | **Character Count Indicator** | **Dữ liệu được mã hóa** |
| 0010 | 000001011 | 01100001011 01111000110 10001011100 10110111000 10011010100 001101 |

**Bước 6: Chia thành các “codeword” 8-bit và thêm Byte đệm nếu cần.**

Sau khi nhận được chuỗi bit bao gồm chỉ báo chế độ, chỉ báo đếm ký tự và bit dữ liệu như được mô tả từ bước 1 đến bước 5, có thể cần phải thêm số 0 và byte đệm, vì đặc tả mã QR yêu cầu rằng chuỗi bit phải lấp đầy hoàn toàn tổng dung lượng của mã QR.

Quá trình thêm số 0 và byte đệm vào chuỗi bit:

**1. Xác định số bit cần thiết cho mã QR:**

Để xác định số lượng bit dữ liệu cần thiết cho một mã QR cụ thể, cần tham khảo bảng sửa lỗi. Tìm phiên bản và mức sửa lỗi đang được sử dụng cho mã QR đang được mã hóa và tìm số trong cột có nhãn "Tổng số từ mã dữ liệu cho phiên bản này và cấp EC". Nhân số này với 8 để có tổng số bit dữ liệu cần thiết cho phiên bản này và mức sửa lỗi.

Ví dụ, theo bảng, mã phiên bản 1-Q có tổng số 13 từ mã dữ liệu. Do đó, tổng số bit cần thiết cho mã QR này là:

13 × 8 = 104 (bit)

**2. Thêm dấu kết thúc Terminator 0 nếu cần thiết**

Nếu chuỗi bit ngắn hơn tổng số bit cần thiết thì phải thêm dấu kết thúc tối đa 4 số 0 vào phía bên phải của chuỗi. Nếu chuỗi bit ngắn hơn bốn bit so với số bit yêu cầu, hãy thêm bốn số 0 vào cuối. Nếu chuỗi bit ngắn hơn bốn bit, chỉ thêm số 0 cần thiết để đạt được số bit yêu cầu.

Ví dụ: nếu mã hóa HELLO WORLD bằng mã QR phiên bản 1-Q thì tổng số bit cần thiết như đã đề cập ở phần trước là 104 bit. Chuỗi bit dữ liệu được hiển thị ở bước 5 dài 74 bit. Dấu kết thúc chỉ được dài tối đa 4 bit, vì vậy phải thêm bốn số 0 vào bên phải chuỗi. Chuỗi kết quả vẫn còn quá ngắn để lấp đầy dung lượng 104 bit, nhưng đặc tả mã QR yêu cầu dấu kết thúc có độ dài tối đa là 4 số 0. Thay vào đó, nếu chuỗi có 102 bit thì dấu kết thúc sẽ chỉ có độ dài 2 bit.

Dưới đây là ví dụ về chuỗi HELLO WORLD có thêm dấu kết thúc:

|  |  |  |  |
| --- | --- | --- | --- |
| **Mode Indicator** | **Character Count Indicator** | **Dữ liệu được mã hóa** | **Dấu kết thúc Terminator** |
| 0010 | 000001011 | 01100001011 01111000110 10001011100 10110111000 10011010100 001101 | 0000 |

**3. Thêm nhiều số 0 để biến độ dài thành bội số của 8**

Sau khi thêm dấu kết thúc Terminator, nếu số bit trong chuỗi không phải là bội số của 8, trước tiên hãy thêm số 0 vào chuỗi bên phải để biến độ dài của chuỗi thành bội số của 8.

Ví dụ: sau khi thêm dấu kết thúc vào chuỗi HELLO WORLD, độ dài trở thành 78 bit. Đây không phải là bội số của 8. Chuỗi bit được hiển thị ở đây được chia thành các byte nhị phân 8 bit:

00100000 01011011 00001011 01111000 11010001 01110010 11011100 01001101 01000011 010000

Có sáu bit ở cuối. Thêm hai số 0 để biến nó thành byte nhị phân 8 bit:

00100000 01011011 00001011 01111000 11010001 01110010 11011100 01001101 01000011 010000**00**

**4. Đệm thêm Byte nếu Chuỗi vẫn quá ngắn**

Nếu chuỗi vẫn chưa đủ dài để lấp đầy dung lượng tối đa, hãy thêm các byte sau vào cuối chuỗi, lặp lại cho đến khi chuỗi đạt độ dài tối đa:

11101100 00010001

Các byte này tương ứng với 236 và 17. Chúng được đặc tả mã QR yêu cầu cụ thể để thêm vào nếu chuỗi bit quá ngắn ở giai đoạn này.

Ví dụ: chuỗi HELLO WORLD ở trên dài 80 bit. Dung lượng cần thiết cho mã 1-Q như đã nêu trước đó là 104 bit. Số bit phải được thêm vào để lấp đầy dung lượng còn lại là 104 - 80 hoặc 24. Chia số này cho 8: 24/8 = 3. Do đó, ba byte đệm pad phải được thêm vào cuối chuỗi dữ liệu.

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

## **Mã hóa sữa lỗi**

Mã sửa lỗi cho phép người đọc mã QR phát hiện và sửa lỗi trong mã QR. Phần này giải thích cách tạo các mã sửa lỗi sau khi **mã hóa dữ liệu**.

**Bước 1: Chia các từ mã dữ liệu thành các khối nếu cần thiết:**

Trước khi tạo từ mã sửa lỗi, có thể cần phải chia từ mã dữ liệu thành các khối nhỏ hơn nếu mã QR lớn hơn “version 2”. Ví dụ: nếu tạo mã QR 5-Q, bảng sửa lỗi cho biết rằng mã 5-Q có 62 “data codeword” (chuỗi số nhị phân 8 bit).

Ví dụ:

(codeword #1) 01000011

(codeword #2) 01010101

(codeword #3) 01000110

(codeword #4) 10000110

(codeword #5) 01010111

(codeword #6) 00100110

(codeword #7) 01010101

(codeword #8) 11000010

(codeword #9) 01110111

(codeword #10) 00110010

(codeword #11) 00000110

(codeword #12) 00010010

(codeword #13) 00000110

(codeword #14) 01100111

(codeword #15) 00100110

(codeword #16) 11110110

(codeword #17) 11110110

(codeword #18) 01000010

(codeword #19) 00000111

(codeword #20) 01110110

(codeword #21) 10000110

(codeword #22) 11110010

(codeword #23) 00000111

(codeword #24) 00100110

(codeword #25) 01010110

(codeword #26) 00010110

(codeword #27) 11000110

(codeword #28) 11000111

(codeword #29) 10010010

(codeword #30) 00000110

(codeword #31) 10110110

(codeword #32) 11100110

(codeword #33) 11110111

(codeword #34) 01110111

(codeword #35) 00110010

(codeword #36) 00000111

(codeword #37) 01110110

(codeword #38) 10000110

(codeword #39) 01010111

(codeword #40) 00100110

(codeword #41) 01010010

(codeword #42) 00000110

(codeword #43) 10000110

(codeword #44) 10010111

(codeword #45) 00110010

(codeword #46) 00000111

(codeword #47) 01000110

(codeword #48) 11110111

(codeword #49) 01110110

(codeword #50) 01010110

(codeword #51) 11000010

(codeword #52) 00000110

(codeword #53) 10010111

(codeword #54) 00110010

(codeword #55) 11100000

(codeword #56) 11101100

(codeword #57) 00010001

(codeword #58) 11101100

(codeword #59) 00010001

(codeword #60) 11101100

(codeword #61) 00010001

(codeword #62) 11101100

Bảng sửa lỗi đề cập đến "nhóm 1" và "nhóm 2", cũng như "số khối". Điều này có nghĩa là các từ mã dữ liệu phải được chia thành tối đa hai nhóm và trong mỗi nhóm, các từ mã dữ liệu có thể được chia thành các khối. Các từ mã dữ liệu được chia ra một cách tuần tự (tức là bắt đầu bằng từ mã 1, sau đó là từ mã 2, v.v.).

Đối với mã 5-Q, nó nói rằng có hai nhóm, nhóm đầu tiên nên được chia thành 2 khối, mỗi khối chứa 15 từ mã dữ liệu và nhóm thứ hai nên được chia thành 2 khối, mỗi khối chứa 16 dữ liệu. các từ mã. Lưu ý rằng 15+15+16+16 = 62, là tổng số từ mã dữ liệu. Dưới đây là các từ mã ở trên, được chia thành các nhóm và khối chính xác để minh họa cách thực hiện bước này.

|  |  |  |
| --- | --- | --- |
| Số nhóm | Chặn số | Từ mã dữ liệu trong nhóm |
| Nhóm 1 | Khối 1 | (codeword #1) 01000011  (codeword #2) 01010101  (codeword #3) 01000110  (codeword #4) 10000110  (codeword #5) 01010111  (codeword #6) 00100110  (codeword #7) 01010101  (codeword #8) 11000010  (codeword #9) 01110111  (codeword #10) 00110010  (codeword #11) 00000110  (codeword #12) 00010010  (codeword #13) 00000110  (codeword #14) 01100111  (codeword #15) 00100110 |
| Khối 2 | (codeword #16) 11110110  (codeword #17) 11110110  (codeword #18) 01000010  (codeword #19) 00000111  (codeword #20) 01110110  (codeword #21) 10000110  (codeword #22) 11110010  (codeword #23) 00000111  (codeword #24) 00100110  (codeword #25) 01010110  (codeword #26) 00010110  (codeword #27) 11000110  (codeword #28) 11000111  (codeword #29) 10010010  (codeword #30) 00000110 |
| Nhóm 2 | Khối 1 | (codeword #31) 10110110  (codeword #32) 11100110  (codeword #33) 11110111  (codeword #34) 01110111  (codeword #35) 00110010  (codeword #36) 00000111  (codeword #37) 01110110  (codeword #38) 10000110  (codeword #39) 01010111  (codeword #40) 00100110  (codeword #41) 01010010  (codeword #42) 00000110  (codeword #43) 10000110  (codeword #44) 10010111  (codeword #45) 00110010  (codeword #46) 00000111 |
| Khối 2 | (codeword #47) 01000110  (codeword #48) 11110111  (codeword #49) 01110110  (codeword #50) 01010110  (codeword #51) 11000010  (codeword #52) 00000110  (codeword #53) 10010111  (codeword #54) 00110010  (codeword #55) 11100000  (codeword #56) 11101100  (codeword #57) 00010001  (codeword #58) 11101100  (codeword #59) 00010001  (codeword #60) 11101100  (codeword #61) 00010001  (codeword #62) 11101100 |

Trong bảng, lưu ý rằng các khối trong nhóm 1 bao gồm 15 từ mã dữ liệu và các khối trong nhóm 2 bao gồm 16 từ mã dữ liệu như được chỉ định trong bảng sửa lỗi . Cũng lưu ý rằng chúng được chia ra một cách tuần tự. Nghĩa là, chúng có cùng thứ tự như trước khi được tách thành các khối.

Bảng sửa lỗi cũng cho biết rằng đối với mã 5-Q, có 18 từ mã sửa lỗi trên mỗi khối. Trong ví dụ này, có bốn khối, do đó sẽ có bốn bộ 18 từ mã sửa lỗi, tổng cộng có 72 từ mã sửa lỗi.

Như được hiển thị trong bảng sửa lỗi , các mã QR nhỏ hơn hoàn toàn không yêu cầu chia nhỏ các từ mã dữ liệu, do đó, đối với mã 1-M (ví dụ), tất cả 16 từ mã dữ liệu sẽ được sử dụng như một khối duy nhất và chỉ cần tạo ra 10 từ mã sửa lỗi.

Trước khi tiếp tục ví dụ 5-Q, cần giải thích các bước cơ bản của việc sửa lỗi Reed-Solomon.

*Tìm hiểu phép chia dài thức*

Các từ mã sửa lỗi sẽ được tạo ra bằng phương pháp gọi là sửa lỗi Reed-Solomon. Một phần của quá trình này là thực hiện phép chia dài đa thức. Tức là chia một đa thức cho một đa thức khác. Hướng dẫn này giả định bạn đã hiểu cách thực hiện phép chia dài tiêu chuẩn (tức là với số chứ không phải đa thức) bằng tay. Để biết thêm thông tin, hãy đọc bài viết của wikiHow về cách thực hiện phép chia dài .

Phép chia dài đa thức phức tạp hơn một chút so với phép chia dài tiêu chuẩn. Để giúp làm rõ quy trình, đây là một ví dụ đơn giản về phép chia dài đa thức.

Ví dụ này cho thấy 3x 2 + x - 1 (số bị chia) chia cho x + 1 (số chia). Những đa thức này lần đầu tiên được đưa vào một hoạt cảnh (xem bài viết về phép chia dài của Wikipedia để được giải thích về ký hiệu này).

\_\_\_\_\_\_\_\_\_\_\_\_\_

x + 1 | 3x 2 + x - 1

Trong mỗi bước của phép chia dài, nhân x + 1 với số nào đó để làm cho số hạng đầu tiên của nó bằng số hạng của đa thức ở cuối bảng.

Đối với bước đầu tiên, 3x 2 + x - 1 nằm ở cuối bảng, vì vậy hãy nhân x + 1 với 3x . Điều này dẫn đến 3x 2 + 3x , có cùng số hạng đầu tiên với số hạng của đa thức ở cuối bảng. Đây là bảng cập nhật.

3x

x + 1 | 3x 2 + x - 1

3x2 + 3x

Bây giờ trừ 3x 2 + 3x từ 3x 2 + x - 1 . Điều này dẫn đến -2x - 1 (vì x - 3x là (1 - 3)x hoặc -2x ). - 1 ở cuối được chuyển từ đa thức ban đầu. Đây là bảng cập nhật.

3x

x + 1 | 3x 2 + x - 1

3x2 + 3x

- 2x - 1

Bây giờ, một lần nữa, nhân x + 1 với một số hạng sẽ thu được một đa thức có số hạng đầu tiên giống với số hạng của đa thức dưới cùng. Trong trường hợp này, hãy thực hiện -2 (x + 1) để nhận được -2x - 2 . Đây là bảng cập nhật.

3x - 2

x + 1 | 3x 2 + x - 1

3x2 + 3x

- 2x - 1

- 2x - 2

Và bây giờ, một lần nữa, hãy trừ hai đa thức. -2x bị loại bỏ, nhưng điều này để lại -1 - 2, bằng 1, như được hiển thị trong hoạt cảnh cuối cùng bên dưới.

3x - 2

x + 1 | 3x 2 + x - 1

3x2 + 3x

- 2x - 1

- 2x - 2

1

Số dư là 1. Về mặt kỹ thuật thì nó là 1x 0 . Cần phải nhân (x + 1) với 1/x để làm cho số hạng đầu tiên giống nhau, nhưng vì đây là một phân số nên có nghĩa là không thể chia thêm được nữa. Số 1 ở cuối bảng là phần còn lại.

Nhìn chung, các bước của phép chia dài đa thức là:

Tìm số hạng thích hợp để nhân số đó với Kết quả của phép nhân phải có cùng số hạng đầu tiên với số bị chia (trong bước nhân đầu tiên) hoặc số dư (trong tất cả các bước nhân tiếp theo).

Trừ kết quả từ số bị chia (trong bước nhân đầu tiên) hoặc số dư (trong tất cả các bước nhân tiếp theo).

Lặp lại bước 1 và 2 cho đến khi không thể nhân với một số nguyên nữa, hay nói cách khác, cần phải nhân với một phân số. Số ở cuối bảng là số dư.

Phép chia dài đa thức cần thiết cho việc sửa lỗi Reed-Solomon đơn giản hơn trong một số cách so với ví dụ này, vì sẽ không cần thiết phải xử lý số mũ của các số hạng đa thức.

*Tìm hiểu trường Galois*

Như đã đề cập ở phần trước, để tạo ra các từ mã sửa lỗi, quá trình này sử dụng một phương pháp gọi là sửa lỗi Reed-Solomon. Cùng với phép chia dài đa thức, phương pháp này sử dụng trường Galois, về cơ bản là một tập hợp số bị hạn chế, cũng như một số phép toán tạo ra các số vẫn nằm trong tập hợp đó.

Tiêu chuẩn Mã QR cho biết sử dụng số học module 2 theo bit và số học module 100011101 theo byte. Điều này có nghĩa là sử dụng Trường Galois 2 8 , hay nói cách khác Trường Galois 256, đôi khi được viết là GF(256).

Các số trong GF(256) đều sẽ nằm trong khoảng từ 0 đến 255 (đã bao gồm). Lưu ý rằng đây là dãy số tương tự có thể được biểu diễn bằng byte 8 bit (byte 8 bit lớn nhất có thể là 11111111, bằng 255).

Điều này có nghĩa là tất cả các phép toán trong GF(256) sẽ dẫn đến các số có thể được biểu diễn dưới dạng byte 8 bit.

*Tìm hiểu phép toán trường Galois*

Như đã đề cập trước đó, GF(256) chứa các số từ 0 đến 255. Các phép toán trong GF(256) có tính chất chu kỳ, nghĩa là nếu một phép toán được thực hiện trong GF(256) dẫn đến một số lớn hơn 255, thì cần phải sử dụng phép toán module để có được một số vẫn ở Cánh đồng Galois.

Trong Trường Galois, số âm có cùng giá trị với số dương, vì vậy -n = n . Nói cách khác, luôn sử dụng giá trị tuyệt đối của các số liên quan đến số học Trường Galois.

Điều này có nghĩa là phép cộng và phép trừ trong Trường Galois là như nhau. Phép cộng và phép trừ trong Trường Galois được thực hiện bằng cách thực hiện phép cộng hoặc phép trừ như bình thường, nhưng sau đó thực hiện phép toán module. Và vì chúng tôi đang sử dụng số học module 2 theo bit (như đã đề cập trong đặc tả mã QR), nên điều này cũng giống như thực hiện thao tác XOR. Ví dụ:

1 + 1 = 2 % 2 = 0

hoặc

1 ^ 1 = 0

và

0 + 1 = 1 % 2 = 1

hoặc

0 ^ 1 = 1

Với mục đích mã hóa mã QR, tất cả phép cộng và trừ trong GF(256) được thực hiện bằng cách XOR hai số với nhau.

*Tạo lũy thừa của 2 bằng Byte-Wise module 100011101*

Tất cả các số trong GF(256) có thể được biểu diễn dưới dạng lũy ​​thừa của 2. Cụ thể, tất cả các số trong GF(256) có thể được biểu diễn dưới dạng 2 n , trong đó 0 8 dường như là quá lớn đối với Trường Galois vì nó bằng 256.

Các lũy thừa của 2 từ 0 đến 8 là:

20 = 1

21 = 2

22 = 4

23 = 8

24 = 16

25 = 32

26 = 64

27 = 128

28 = 256

Đặc tả mã QR cho biết sử dụng số học module 100011101 theo byte (trong đó 100011101 là số nhị phân tương đương với 285 ở dạng thập phân). Điều này có nghĩa là khi một số lớn hơn 256 thì nó phải được XOR với 285. Điều này dẫn đến các giá trị không mong muốn cho 28 và lớn hơn.

Nói cách khác:

28 = 256^285 = 29

Lưu ý rằng khi tiếp tục đến 29 , không lấy giá trị thông thường của nó là 512 và XOR với 285 (điều này dù sao cũng sẽ dẫn đến một số quá lớn). Thay vào đó, vì 29 = 28 × 2, hãy sử dụng giá trị 28 đã được tính ở bước trước.

Nói cách khác:

29 = 28 × 2 = 29 × 2 = 58

Tiếp tục sử dụng lũy ​​thừa 2 trước đó để tạo các lũy thừa tiếp theo của 2:

210 = 29 × 2 = 58 × 2 = 116

211 = 210 × 2 = 116 × 2 = 232

Bất cứ khi nào thu được giá trị lớn hơn hoặc bằng 256, một lần nữa XOR với 285:

212 = 211 × 2 = 232 × 2 = 464 ^ 285 = 205

Nói chung, các giá trị luôn bằng 2 lần lũy thừa trước đó và nếu giá trị đó lớn hơn 256 thì được XOR với 285.

Sử dụng quy trình này, tất cả các số trong GF(256) có thể được biểu diễn bằng 2n , trong đó n là một số trong phạm vi 0

*Tìm hiểu phép nhân với log và antilog*

Bởi vì tất cả các giá trị có thể được biểu diễn bằng 2n như đã giải thích ở trên, nên có thể ghi nhật ký và phản log để đơn giản hóa phép nhân trong GF(256). (Ngẫu nhiên, đây là điều mà các quy tắc trượt thực hiện, mặc dù chúng không bị hạn chế ở GF(256)).

Việc đơn giản hóa là có thể thực hiện được bởi vì, nói chung (tức là không chỉ trong Trường Galois), người ta có thể nhân hai số p và q bằng phép toán sau:

b(logbp + logbq)

Trong trường hợp này, cơ số 2 được sử dụng nên phép toán trên trở thành

2(log2p + log2q)

Và phép toán log2x cho chúng ta biết nhân 2 bao nhiêu lần để có kết quả là x. Nói cách khác, nó xác định n trong đó 2n = x.

Điều này cung cấp một lối tắt để nhân các số trong GF(256). Khi nhân hai số có cùng cơ số, chẳng hạn như 22 × 28, điều này tương đương với việc cộng hai số mũ với nhau, như sau:

22 × 28 = 2(2 + 8) = 210

Điều này có nghĩa là vì tất cả các giá trị của GF(256) đã được biểu diễn dưới dạng lũy ​​thừa của hai (đã giải thích ở phần trước), nên mọi phép nhân trong GF(256) có thể được thực hiện dưới dạng phép cộng số mũ cơ số 2.

Ví dụ: để nhân 16 × 32, điều này giống như nhân 24 với 25 . Như đoạn trước đã giải thích, số này giống với 2(4 + 5) hoặc 29 . Giá trị của 29 đã được tính toán, do đó, trong GF(256) với module 285 theo byte, 16 × 32 là 29 , được xác định trong phần trước bằng 58.

Khi cộng số mũ, nếu số mũ lớn hơn hoặc bằng 256 thì chỉ cần áp dụng module 255. Nói cách khác:

2170 × 2164 = 2(170+164) = 2334 → 2334 % 255 = 279.

Do đó, tất cả những gì cần thiết để thực hiện phép nhân trong GF(256) là tạo ra tất cả các lũy thừa của 2. Các giá trị này đã được tính toán và có thể tìm thấy trong bảng log antilog . Bảng sử dụng ký hiệu alpha, trong đó α = 2. Đặc tả Mã QR cũng sử dụng ký hiệu alpha trong Phụ lục A.

*Tìm hiểu đa thức phát (Generator Polynomial)*

Chúng tôi đã đạt được nhiều tiến bộ trong việc tạo ra các từ mã sửa lỗi, nhưng vẫn chưa đạt đến mức đó. Bước tiếp theo là tìm hiểu các đa thức sinh.

Như đã đề cập trước đó, mã hóa sửa lỗi sử dụng phép chia dài đa thức. Để làm điều đó, cần có hai đa thức. Đa thức đầu tiên được sử dụng được gọi là **đa thức thông điệp** . Đa thức thông điệp sử dụng các từ mã dữ liệu từ bước mã hóa dữ liệu làm hệ số của nó. Ví dụ: nếu các từ mã dữ liệu, được chuyển đổi thành số nguyên, là 25, 218 và 35 thì đa thức thông báo sẽ là 25x2 + 218x + 35. Trong thực tế, đa thức thông báo thực cho mã QR tiêu chuẩn dài hơn nhiều, nhưng đây chỉ là một ví dụ.

Đa thức thông báo sẽ được chia cho **đa thức sinh** . Đa thức tạo là đa thức được tạo bằng cách nhân

(x - α0 ) ... (x - αn-1 )

trong đó n là số lượng từ mã sửa lỗi phải được tạo ra (xem bảng sửa lỗi ). Như đã đề cập ở phần trước, α (alpha) bằng 2.

Có thể lập trình tạo tất cả các đa thức sinh có thể cần thiết. Trong quá trình này, hướng dẫn này sẽ sử dụng ký hiệu alpha và phép nhân log antilog đã được mô tả trước đó trên trang này.

Đặc tả mã QR liệt kê các đa thức tạo trong Phụ lục A, bắt đầu bằng 2 và kết thúc bằng 68. Mặc dù mã QR tiêu chuẩn sẽ luôn yêu cầu nhiều hơn 2 từ mã sửa lỗi trên mỗi khối, trang này sẽ hiển thị cách tính đa thức tạo cho 2 từ mã sửa lỗi , bởi vì nó minh họa quá trình tính toán tất cả phần còn lại của đa thức sinh.

*Đa thức tạo cho 2 từ mã sửa lỗi*

Đầu tiên, nhân (x - α0 ) và (x - α1 ).

Vì hệ số của x là 1 và vì α 0 = 1 nên điều này có thể được viết là

(α0x - α0 ) × (α0x - α1 )

Nhân mỗi số hạng của phần thứ nhất với mỗi số hạng của phần thứ hai để có kết quả:

(α0x1 × α0x1 ) + (α0x0 × α0x1 ) + (α0x1 × α1x0 ) + (α0x0 × α1x0 )

Lưu ý rằng phép cộng số mũ có thể được sử dụng ở đây để thực hiện các phép nhân:

(α(0+0)x(1+1)) + (α(0 +0)x(0+1)) + (α(0+1) x(1+0)) + (α(0+1)x(0+0))

Kết quả:

α0x2 + α0x1 + α1x1 + α1x0

Bây giờ hãy kết hợp các thuật ngữ giống nhau. Có hai số hạng x 1 nên hãy cộng chúng lại với nhau.

α0x2 + (α0 + α1)x1 + α1x0

Hãy nhớ rằng phép cộng trong GF(256) được thực hiện bởi XOR. Chuyển đổi các bảng chữ cái thành các bản sao số nguyên của chúng bằng cách sử dụng bảng log antilog , sau đó thực hiện XOR.

x2 + (12)x1 + 2x0

x2 + 3x1 + 2x0

Quay lại ký hiệu alpha, đây là

α0x2 + α25x1 + α1x0

Đây là đa thức tạo cho 2 từ mã sửa lỗi.

*Đa thức tạo cho 3 từ mã sửa lỗi*

Tất cả các đa thức sinh còn lại có thể được tạo theo cách tương tự, sử dụng đa thức sinh từ mỗi bước trước đó. Bắt đầu với đa thức sinh trước đó và nhân nó với thừa số tiếp theo, trong trường hợp này là (x - α2)

(α0x2 + α25x1 + α1x0) × (α0x1 + α2x0)

(α0x2 × α0x1) + (α25x1 × α0x1) + (α1x0 × α0x1) + (α0x2 × α2x0) + (α25x1 × α2x0) + (α1x0 × α2x0)

Để nhân, cộng các số mũ lại với nhau như sau:

(α(0+0)x(2+1)) + (α(25+0)x(1+1)) + (α(1+0)x(0 +1)) + (α(0+2)x(2+0)) + (α(25+2)x(1+0)) + (α(1+2)x(0+0))

Sau khi cộng số mũ ta được kết quả:

α0x3 + α25x2 + α1x1 + α2x2 + α27x1 + α3x0

Bây giờ kết hợp các thuật ngữ tương tự.

α0x3 + (α25 ^ α2)x2 + (α1 ^ α27)x1 + α3x0

α0x3 + (34)x2 + (212)x1 + α3x0

α0x3 + 7x2 + 14x1 + α3x0

α0x3 + α198x2 + α199x1 + α3x0

Đây là đa thức tạo cho 3 từ mã sửa lỗi.

*Khi số mũ lớn hơn hoặc bằng 256*

Khi thực hiện phép nhân bằng cách cộng các số mũ lại với nhau, đôi khi điều này có thể dẫn đến số mũ lớn hơn hoặc bằng 256. Trong trường hợp này, hãy áp dụng module 255 TRƯỚC KHI kết hợp các số hạng tương tự. Ví dụ:

α257x4 = α(257 % 255)x4 = α2x4

*Các đa thức sinh khác*

Nói chung, có thể tạo tất cả các đa thức sinh g(x) như minh họa dưới đây:

(g(x) cho 2 từ mã sửa lỗi) × (x - α2 ) = (g(x) cho 3 từ mã sửa lỗi)

(g(x) cho 3 từ mã sửa lỗi) × (x - α3 ) = (g(x) cho 4 từ mã sửa lỗi)

(g(x) cho 4 từ mã sửa lỗi) × (x - α4 ) = ( g(x) cho 5 từ mã sửa lỗi)

...

(g(x) cho n-1 từ mã sửa lỗi) × (x - αn-1 ) = (g(x) cho n từ mã sửa lỗi)

Để có được một đa thức tạo cụ thể mà không cần phải tính toán theo chương trình, công cụ tạo đa thức có thể tạo ra nó.

*Tạo từ mà sửa lỗi*

Bây giờ cuối cùng đã đến lúc bắt đầu tạo từ mã sửa lỗi. Phần này sử dụng các từ mã dữ liệu từ ví dụ HELLO WORLD trên trang mã hóa chế độ chữ và số . Những từ mã này sẽ được sử dụng làm hệ số của đa thức thông điệp. Ví dụ này sử dụng mức sửa lỗi M và phiên bản 1 để tạo mã 1-M.

*Đa thức tin nhắn*

Bước mã hóa dữ liệu dẫn đến các từ mã dữ liệu sau cho HELLO WORLD dưới dạng mã 1-M.

00100000 01011011 00001011 01111000 11010001 01110010 11011100 01001101 01000011 01000000 11101100 00010001 11101100 0001 0001 11101100 00010001

Chuyển các số nhị phân đó thành số thập phân:

32, 91, 11, 120, 209, 114, 220, 77, 67, 64, 236, 17, 236, 17, 236, 17

Những con số này là hệ số của đa thức thông điệp . Nói cách khác:

32x15 + 91x14 + 11x13 ... v.v.

*Đa thức sinh*

Đầu tiên, lấy đa thức sinh. Vì đây là mã 1-M nên bảng sửa lỗi sẽ tạo ra 10 từ mã sửa lỗi. Do đó, hãy sử dụng đa thức tạo sau:

x10 + α251x9 + α67x8 + α46x7 + α61x6 + α118x5 + α70x4 + α64x3 + α94x2 + α32x + α45

(Tham khảo công cụ tạo đa thức và phần thảo luận về đa thức tạo trước đó trên trang này.)

*Các bước chia*

Bây giờ là lúc chia đa thức thông điệp cho đa thức sinh. Việc này được thực hiện theo cách tương tự như phép chia dài đa thức được trình bày trước đó trên trang. Dưới đây là các bước chia, được cập nhật để tính đến số học Trường Galois:

Tìm số hạng thích hợp để nhân đa thức sinh với. Kết quả của phép nhân phải có cùng số hạng đầu tiên với đa thức thông báo (trong bước nhân đầu tiên) hoặc phần dư (trong tất cả các bước nhân tiếp theo).

XOR kết quả với đa thức thông báo (trong bước nhân đầu tiên) hoặc phần dư (trong tất cả các bước nhân tiếp theo).

Thực hiện các bước này n lần, trong đó n là số lượng từ mã dữ liệu.

Lưu ý sự khác biệt giữa các bước này và các bước của phép chia dài đa thức thông thường. Thay vì trừ sau bước nhân, chúng ta thực hiện phép XOR (trong GF(256), cũng giống như vậy).

Quan trọng hơn, sau khi chia hai đa thức sẽ có số dư. Các hệ số của phần còn lại này là các từ mã sửa lỗi.

Việc phân chia sẽ được minh họa từng bước trong phần tiếp theo.

*Chia đa thức thông báo cho đa thức tạo*

Bước đầu tiên của phép chia là chuẩn bị đa thức thông điệp cho phép chia. Đa thức thông điệp đầy đủ là:

32x15 + 91x14 + 11x13 + 120x12 + 209x11 + 114x10 + 220x9 + 77x8 + 67x7 + 64x6 + 236x5 + 17x4 + 236x3 + 17x2 + 236x1 + 17

Để đảm bảo rằng số mũ của số hạng dẫn đầu không trở nên quá nhỏ trong quá trình chia, hãy nhân đa thức thông báo với x n trong đó n là số lượng từ mã sửa lỗi cần thiết. Trong trường hợp này n là 10, đối với 10 từ mã sửa lỗi, nhân đa thức thông báo với x10 , ta được:

32x25 + 91x24 + 11x23 + 120x 22 + 209x21 + 114x20 + 220x19 + 77x18 + 67x17 + 64x16 + 236x15 + 17x14 + 236x13 + 17x12 + 236x11 + 17x10

Số hạng đầu của đa thức sinh cũng phải có cùng số mũ, do đó nhân với x15 để có được

α0x25 + α251x24 + α67x23 + α46x22 + α61x21 + α118x20 + α70x19 + α64x18 + α94x17 + α32x16 + α45x15

Bây giờ có thể thực hiện các bước chia lặp đi lặp lại. Số bước trong phép chia phải bằng số số hạng trong đa thức thông điệp. Trong trường hợp này, việc chia sẽ mất 16 bước để hoàn thành. Điều này sẽ dẫn đến phần còn lại có 10 số hạng. Những thuật ngữ này sẽ là 10 từ mã sửa lỗi được yêu cầu.

*Bước 1a: Nhân đa thức tạo với số hạng đầu của đa thức thông báo*

Bước đầu tiên là nhân đa thức sinh với số hạng đầu của đa thức thông báo. Số hạng dẫn đầu trong trường hợp này là 32x 25 . Vì ký hiệu alpha giúp thực hiện phép nhân dễ dàng hơn nên bạn nên chuyển đổi ký hiệu 32x25 thành ký hiệu alpha. Theo bảng log antilog, đối với giá trị nguyên 32, số mũ alpha là 5. Do đó 32 = α5 . Nhân đa thức sinh với α5 :

Số mũ của các chữ cái được cộng lại với nhau. Trong trường hợp này, ít nhất một trong các số mũ lớn hơn 255, do đó hãy thực hiện module 255 như sau:

α5x25 + α(256 % 255)x24 + α72x23 + α51x22 + α66x21 + α123x20 + α75x19 + α69x18 + α99x17+ α37x16 + α50x15

Kết quả là:

α5x25 + α1x24 + α72x23 + α51x22 + α66x21 + α123x20 + α75x19 + α69x18 + α99x17 + α37x16 + α50x15

Bây giờ, hãy chuyển đổi ký hiệu này thành ký hiệu số nguyên:

32x25 + 2x24 + 101x23 + 10x22 + 97x21 + 197x20 + 15x19 + 47x18 + 134x17 + 74x16 + 5x15

*Bước 1b: XOR kết quả với đa thức thông điệp*

Vì đây là bước chia đầu tiên nên XOR kết quả từ 1a với đa thức thông điệp.

(32 ⊕ 32)x25 + (91 ⊕ 2)x24 + (11 ⊕ 101)x23 + (120 ⊕ 10)x22 + (209 ⊕ 97)x21 + (114 ⊕ 197)x20 + (220 ⊕ 15)x19 + (77 ⊕ 47)x18 + (67 ⊕ 134)x17 + (64 ⊕ 74)x16 + (236 ⊕ 5)x15 + (17 ⊕ 0)x14 + (236 ⊕ 0 )x13 + (17 ⊕ 0)x12 + (236 ⊕ 0)x11 + (17 ⊕ 0)x10

Kết quả là:

0x25 + 89x24 + 110x23 + 114x22 + 176x21 + 183x20 + 211x19 + 98x18 + 197x17 + 10x16 + 233x15 + 17x14 + 236x13 + 17x12 + 236x11 + 17x10

Loại bỏ số hạng dẫn đầu 0 để có được:

89x24 + 110x23 + 114x22 + 176x21 + 183x20 + 211x19 + 98x18 + 197x17 + 10x16 + 233x15 + 17x14 + 236x13 + 17x12 + 236x11 + 17x10

*Bước 2a: Nhân Đa thức tạo với Số hạng dẫn của kết quả XOR từ bước trước*

Tiếp theo, nhân đa thức tạo với số hạng dẫn đầu của kết quả XOR ở bước trước. Số hạng dẫn đầu trong trường hợp này là 89x 24 . Chuyển đổi ký hiệu 89x24 sang alpha. Theo bảng log antilog, đối với giá trị số nguyên 89, số mũ alpha là 210. Do đó 89 = α210 . Nhân đa thức sinh với α210:

(α210 × α0)x24 + (α210 × α251)x23 + (α210 × α67)x22 + (α210 × α46)x21 + (α210 × α61)x20 + (α210 × α118)x19 + (α210 × α70)x18 + (α210 × α64)x17 + (α210 × α94)x16 + (α210 × α32)x15 + (α210 × α45)x14

Số mũ của các chữ cái được cộng lại với nhau. Trong trường hợp này, ít nhất một trong các số mũ lớn hơn 255, do đó hãy thực hiện module 255 như sau:

α210x24 + α(461 % 255)x23 + α(277 % 255)x22 + α(256 % 255)x21 + α(271 % 255)x20 + α(328 % 255)x19 + α(280 % 255)x18 + α(274 % 255)x17 + α(304 % 255)x16 + α242x15 + α255x14

Kết quả là:

α210x24 + α206x23 + α22x22 + α1x21 + α16x20 + α73x19 + α25x18 + α19x17 + α49x16 + α242x15 + α255x14

Bây giờ, hãy chuyển đổi ký hiệu này thành ký hiệu số nguyên:

89x24 + 83x23 + 234x22 + 2x21 + 76x20 + 202x19 + 3x18 + 90x17 + 140x16 + 176x15 + 1x14

*Bước 2b: XOR kết quả với kết quả ở bước 1b*

Sử dụng kết quả từ bước 1b để thực hiện phép XOR tiếp theo.

(89 ⊕ 89)x24 + (110 ⊕ 83)x23 + (114 ⊕ 234)x22 + (176 ⊕ 2)x21 + (183 ⊕ 76)x20 + (211 ⊕ 202)x19 + (98 ⊕ 3)x18 + (197 ⊕ 90)x17 + (10 ⊕ 140)x16 + (233 ⊕ 176)x15 + (17 ⊕ 1)x14 + (236 ⊕ 0)x13 + (17 ⊕ 0 )x12 + (236 ⊕ 0)x11 + (17 ⊕ 0)x10

Kết quả là:

0x24 + 61x23 + 152x22 + 178x21 + 251x20 + 25x19 + 97x18 + 159x17 + 134x16 + 89x15 + 16x14 + 236x13 + 17x12 + 236x11 + 17x10

Loại bỏ số hạng dẫn đầu 0 để có được:

61x23 + 152x22 + 178x21 + 251x20 + 25x19 + 97x18 + 159x17 + 134x16 + 89x15 + 16x14 + 236x13 + 17x12 + 236x11 + 17x10

*Bước 3a: Nhân Đa thức tạo với Số hạng dẫn của kết quả XOR từ bước trước*

Tiếp theo, nhân đa thức tạo với số hạng dẫn đầu của kết quả XOR ở bước trước. Số hạng dẫn đầu trong trường hợp này là 61x23 . Chuyển đổi ký hiệu 61x23 sang alpha. Theo bảng log antilog, đối với giá trị số nguyên 61, số mũ alpha là 228. Do đó 61 = α228 . Nhân đa thức sinh với α228 :

(α228 × α0)x23 + (α228 × α251)x22 + (α228 × α67)x21 + (α228 × α46 )x20 + (α228 × α61)x19 + (α228 × α118)x18 + (α228 × α70)x17 + (α228 × α64)x16 + (α228 × α94 )x15 + (α228 × α32)x14 + (α228 × α45)x13

Số mũ của các chữ cái được cộng lại với nhau. Trong trường hợp này, ít nhất một trong các số mũ lớn hơn 255, do đó hãy thực hiện module 255 như sau:

α228x23 + α(479 % 255)x22 + α(295 % 255)x21 + α(274 % 255)x20 + α(289 % 255)x19 + α(346 % 255)x18 + α(298 % 255)x17 + α(292 % 255)x16 + α(322 % 255)x15 + α(260 % ​​255)x14 + α(273 % 255)x13

Kết quả là:

α228x23 + α224x22 + α40x21 + α19x20 + α34x19 + α91x18 + α43x17 + α37x16 + α67x15 + α5x14 + α18x13

Bây giờ, hãy chuyển đổi ký hiệu này thành ký hiệu số nguyên:

61x23 + 18x22 + 106x21 + 90x20 + 78x19 + 163x18 + 119x17 + 74x16 + 194x15 + 32x14 + 45x13

*Bước 3b: XOR kết quả với kết quả ở bước 2b*

Sử dụng kết quả từ bước 2b để thực hiện phép XOR tiếp theo.

(61 ⊕ 61)x23 + (152 ⊕ 18)x22 + (178 ⊕ 106)x21 + (251 ⊕ 90)x20 + (25 ⊕ 78)x19 + (97 ⊕ 163)x18 + (159 ⊕ 119)x17 + (134 ⊕ 74)x16 + (89 ⊕ 194)x15 + (16 ⊕ 32)x14 + (236 ⊕ 45)x13 + (17 ⊕ 0)x12 + (236 ⊕ 0 )x11 + (17 ⊕ 0)x10

Kết quả là:

0x23 + 138x22 + 216x21 + 161x20 + 87x19 + 194x18 + 232x17 + 204x16 + 155x15 + 48x14 + 193x13 + 17x12 + 236x11 + 17x10

Loại bỏ số hạng dẫn đầu 0 để có được:

138x22 + 216x21 + 161x20 + 87x19 + 194x18 + 232x17 + 204x16 + 155x15 + 48x14 + 193x13 + 17x12 + 236x11 + 17x10

*Bước 4a: Nhân Đa thức tạo với Số hạng dẫn của kết quả XOR từ bước trước*

Tiếp theo, nhân đa thức tạo với số hạng dẫn đầu của kết quả XOR ở bước trước. Số hạng dẫn đầu trong trường hợp này là 138x22 . Chuyển đổi ký hiệu 138x22 sang alpha. Theo bảng log antilog, đối với giá trị số nguyên 138, số mũ alpha là 222. Do đó 138 = α222. Nhân đa thức sinh với α222:

(α222 × α0)x22 + (α222 × α251)x21 + (α222 × α67)x20 + (α222 × α46)x19 + (α 222 × α61)x18 + (α222 × α118)x17 + (α222 × α70)x16 + (α222 × α64)x15 + (α222 × α94)x14 + (α222 × α32 )x13 + (α222 × α45 )x12

Số mũ của các chữ cái được cộng lại với nhau. Trong trường hợp này, ít nhất một trong các số mũ lớn hơn 255, do đó hãy thực hiện module 255 như sau:

α222x22 + α(473 % 255)x21 + α(289 % 255)x20 + α(268 % 255)x19 + α(283 % 255)x18 + α(340 % 255)x17 + α(292 % 255)x16 + α(286 % 255)x15 + α(316 % 255)x14 + α254x13 + α(267 % 255)x12

Kết quả là:

α222x22 + α218x21 + α34x20 + α13x19 + α28x18 + α85x17 + α37x16 + α31x15 + α61x14 + α254x13 + α12x12

Bây giờ, hãy chuyển đổi ký hiệu này thành ký hiệu số nguyên:

138x22 + 43x21 + 78x20 + 135x19 + 24x18 + 214x17 + 74x16 + 192x15 + 111x14 + 142x13 + 205x12

*Bước 4b: XOR kết quả với kết quả ở bước 3b*

Sử dụng kết quả từ bước 3b để thực hiện phép XOR tiếp theo.

(138 ⊕ 138)x22 + (216 ⊕ 43)x21 + (161 ⊕ 78)x20 + (87 ⊕ 135)x19 + (194 ⊕ 24)x18 + (232 ⊕ 214)x17 + (204 ⊕ 74)x16 + (155 ⊕ 192)x15 + (48 ⊕ 111)x14 + (193 ⊕ 142)x13 + (17 ⊕ 205)x12 + (236 ⊕ 0)x11 + (17 ⊕ 0 )x10

Kết quả là:

0x22 + 243x21 + 239x20 + 208x19 + 218x18 + 62x17 + 134x16 + 91x15 + 95x14 + 79x13 + 220x12 + 236x11 + 17x10

Loại bỏ số hạng dẫn đầu 0 để có được:

243x21 + 239x20 + 208x19 + 218x18 + 62x17 + 134x16 + 91x15 + 95x14 + 79x13 + 220x12 + 236x11 + 17x10

*Bước 5a: Nhân Đa thức tạo với Số hạng dẫn của kết quả XOR từ bước trước*

Tiếp theo, nhân đa thức tạo với số hạng dẫn đầu của kết quả XOR ở bước trước. Số hạng dẫn đầu trong trường hợp này là 243x21 . Chuyển đổi 243x21 sang ký hiệu alpha. Theo bảng log antilog, đối với giá trị nguyên 243, số mũ alpha là 233. Do đó 243 = α233 . Nhân đa thức sinh với α233:

(α233 × α0)x21 + (α233 × α251)x20 + (α233 × α67)x19 + (α233 × α46)x18 + (α233 × α61)x17 + (α233 × α118)x16 + (α233 × α70)x15 + (α233 × α64)x14 + (α233 × α94 )x13 + (α233 × α32)x12 + (α233 × α45)x11

Số mũ của các chữ cái được cộng lại với nhau. Trong trường hợp này, ít nhất một trong các số mũ lớn hơn 255, do đó hãy thực hiện module 255 như sau:

α233x21 + α(484 % 255)x20 + α(300 % 255)x19 + α(279 % 255)x18 + α(294 % 255)x17 + α(351 % 255)x16 + α(303 % 255)x15 + α(297 % 255)x14 + α(327 % 255)x13 + α(265 % 255)x12 + α(278 % 255)x11

Kết quả là:

α233x21 + α229x20 + α45x19 + α24x18 + α39x17 + α96x16 + α48x15 + α42x14 + α72x13 + α10x12 + α23x11

Bây giờ, hãy chuyển đổi ký hiệu này thành ký hiệu số nguyên:

243x21 + 122x20 + 193x19 + 143x18 + 53x17 + 217x16 + 70x15 + 181x14 + 101x13 + 116x12 + 201x11

*Bước 5b: XOR kết quả với kết quả ở bước 4b*

Sử dụng kết quả từ bước 4b để thực hiện phép XOR tiếp theo.

(243 ⊕ 243)x21 + (239 ⊕ 122)x20 + (208 ⊕ 193)x19 + (218 ⊕ 143)x18 + (62 ⊕ 53)x17 + (134 ⊕ 217)x16 + (91 ⊕ 70)x15 + (95 ⊕ 181)x14 + (79 ⊕ 101)x13 + (220 ⊕ 116)x12 + (236 ⊕ 201)x11 + (17 ⊕ 0)x10

Kết quả là:

0x21 + 149x20 + 17x19 + 85x18 + 11x17 + 95x16 + 29x15 + 234x14 + 42x13 + 168x12 + 37x11 + 17x10

Loại bỏ số hạng dẫn đầu 0 để có được:

149x20 + 17x19 + 85x18 + 11x17 + 95x16 + 29x15 + 234x14 + 42x13 + 168x12 + 37x11 + 17x10

*Bước 6a: Nhân Đa thức tạo với Số hạng dẫn của kết quả XOR từ bước trước*

Tiếp theo, nhân đa thức tạo với số hạng dẫn đầu của kết quả XOR ở bước trước. Số hạng dẫn đầu trong trường hợp này là 149x20 . Chuyển đổi ký hiệu 149x20 sang alpha. Theo bảng log antilog, đối với giá trị số nguyên 149, số mũ alpha là 184. Do đó 149 = α184. Nhân đa thức sinh với α184:

(α184 × α0)x20 + (α184 × α251)x19 + (α184 × α67)x18 + (α184 × α46)x17 + (α184 × α61)x16 + (α184 × α118)x15 + (α184 × α70)x14 + (α184 × α64)x13 + (α184 × α94)x12 + (α184 × α32)x11 + (α184 × α45)x10

Số mũ của các chữ cái được cộng lại với nhau. Trong trường hợp này, ít nhất một trong các số mũ lớn hơn 255, do đó hãy thực hiện module 255 như sau:

α184x20 + α(435 % 255)x19 + α251x18 + α230x17 + α245x16 + α(302 % 255)x15 + α254x14 + α248x13 + α(278 % 255)x12 + α216x11 + α229x10

Kết quả là:

α184x20 + α180x19 + α251x18 + α230x17 + α245x16 + α47x15 + α254x14 + α248x13 + α23x12 + α216x11 + α229x10

Bây giờ, hãy chuyển đổi ký hiệu này thành ký hiệu số nguyên:

149x20 + 150x19 + 216x18 + 244x17 + 233x16 + 35x15 + 142x14 + 27x13 + 201x12 + 195x11 + 122x10

*Bước 6b: XOR kết quả với kết quả ở bước 5b*

Sử dụng kết quả từ bước 5b để thực hiện XOR tiếp theo.

(149 ⊕ 149)x20 + (17 ⊕ 150)x19 + (85 ⊕ 216)x18 + (11 ⊕ 244)x17 + (95 ⊕ 233)x16 + (29 ⊕ 35)x15 + (234 ⊕ 142)x14 + (42 ⊕ 27)x13 + (168 ⊕ 201)x12 + (37 ⊕ 195)x11 + (17 ⊕ 122)x10

Kết quả là:

0x20 + 135x19 + 141x18 + 255x17 + 182x16 + 62x15 + 100x14 + 49x13 + 97x12 + 230x11 + 107x10

Loại bỏ số hạng dẫn đầu 0 để có được:

135x19 + 141x18 + 255x17 + 182x16 + 62x15 + 100x14 + 49x13 + 97x12 + 230x11 + 107x10

*Bước 7a: Nhân Đa thức tạo với Số hạng dẫn của kết quả XOR ở bước trước*

Tiếp theo, nhân đa thức tạo với số hạng dẫn đầu của kết quả XOR ở bước trước. Số hạng dẫn đầu trong trường hợp này là 135x19 . Chuyển đổi ký hiệu 135x19 sang alpha. Theo bảng log antilog, đối với giá trị số nguyên 135, số mũ alpha là 13. Do đó 135 = α13 . Nhân đa thức sinh với α13:

(α13 × α0)x19 + (α13 × α251)x18 + (α13 × α67)x17 + (α13 × α46)x16 + (α13 × α61)x15 + (α13 × α118)x14 + (α13 × α70)x13 + (α13 × α64)x12 + (α13 × α94)x11 + (α13 × α32)x10 + (α13 × α45)x9

Số mũ của các chữ cái được cộng lại với nhau. Trong trường hợp này, ít nhất một trong các số mũ lớn hơn 255, do đó hãy thực hiện module 255 như sau:

α13x19 + α(264 % 255)x18 + α80x17 + α59x16 + α74x15 + α131x14 + α83x13 + α77x12 + α107x11 + α45x10 + α58x9

Kết quả là:

α13x19 + α9x18 + α80x17 + α59x16 + α74x15 + α131x14 + α83x13 + α77x12 + α107x11 + α45x10 + α58x9

Bây giờ, hãy chuyển đổi ký hiệu này thành ký hiệu số nguyên:

135x19 + 58x18 + 253x17 + 210x16 + 137x15 + 92x14 + 187x13 + 60x12 + 104x11 + 193x10 + 105x9

*Bước 7b: XOR kết quả với kết quả ở bước 6b*

Sử dụng kết quả từ bước 6b để thực hiện phép XOR tiếp theo.

(135 ⊕ 135)x19 + (141 ⊕ 58)x18 + (255 ⊕ 253)x17 + (182 ⊕ 210)x16 + (62 ⊕ 137)x15 + (100 ⊕ 92)x14 + (49 ⊕ 187)x13 + (97 ⊕ 60)x12 + (230 ⊕ 104)x11 + (107 ⊕ 193)x10 + (0 ⊕ 105)x9

Kết quả là:

0x19 + 183x18 + 2x17 + 100x16 + 183x15 + 56x14 + 138x13 + 93x12 + 142x11 + 170x10 + 105x9

Loại bỏ số hạng dẫn đầu 0 để có được:

183x18 + 2x17 + 100x16 + 183x15 + 56x14 + 138x13 + 93x12 + 142x11 + 170x10 + 105x9

*Bước 8a: Nhân Đa thức tạo với Số hạng dẫn của kết quả XOR ở bước trước*

Tiếp theo, nhân đa thức tạo với số hạng dẫn đầu của kết quả XOR ở bước trước. Số hạng dẫn đầu trong trường hợp này là 183x18 . Chuyển đổi ký hiệu 183x18 sang alpha. Theo bảng log antilog, đối với giá trị số nguyên 183, số mũ alpha là 158. Do đó 183 = α158 . Nhân đa thức sinh với α158 :

(α158 × α0)x18 + (α158 × α251)x17 + (α158 × α67)x16 + (α158 × α46)x15 + (α158 × α61)x14 + (α158 × α118)x13 + (α158 × α70)x12 + (α158 × α64)x11 + (α158 × α94)x10 + (α158 × α32)x9 + (α158 × α45)x8

Số mũ của các chữ cái được cộng lại với nhau. Trong trường hợp này, ít nhất một trong các số mũ lớn hơn 255, do đó hãy thực hiện module 255 như sau:

α158x18 + α(409 % 255)x17 + α225x16 + α204x15 + α219x14 + α(276 % 255)x13 + α228x12 + α222x11 + α252x10 + α190x9 + α203x8

Kết quả là:

α 158 x 18 + α 154 x 17 + α 225 x 16 + α 204 x 15 + α 219 x 14 + α 21 x 13 + α 228 x 12 + α 222 x 11 + α 252 x 10 + α 190 x 9 + α 203 x 8

Bây giờ, hãy chuyển đổi ký hiệu này thành ký hiệu số nguyên:

183x18 + 57x17 + 36x16 + 221x15 + 86x14 + 117x13 + 61x12 + 138x11 + 173x10 + 174x9 + 224x8 \_

*Bước 8b: XOR kết quả với kết quả ở bước 7b*

Sử dụng kết quả từ bước 7b để thực hiện phép XOR tiếp theo.

(183 ⊕ 183)x18 + (2 ⊕ 57)x17 + (100 ⊕ 36)x16 + (183 ⊕ 221)x15 + (56 ⊕ 86)x14 + (138 ⊕ 117)x13 + (93 ⊕ 61)x12 + (142⊕ 138)x11 + (170 ⊕ 173)x10 + (105 ⊕ 174)x9 + (0 ⊕ 224)x8

Kết quả là:

0x18 + 59x17 + 64x16 + 106x15 + 110x14 + 255x13 + 96x12 + 4x11 + 7x10 + 199x9 + 224x8

Loại bỏ số hạng dẫn đầu 0 để có được:

59x17 + 64x16 + 106x15 + 110x14 + 255x13 + 96x12 + 4x11 + 7x10 + 199x9 + 224x8

*Bước 9a: Nhân Đa thức tạo với Số hạng dẫn của kết quả XOR ở bước trước*

Tiếp theo, nhân đa thức tạo với số hạng dẫn đầu của kết quả XOR ở bước trước. Số hạng dẫn đầu trong trường hợp này là 59x17. Chuyển đổi ký hiệu 59x17 sang alpha. Theo bảng log antilog, đối với giá trị số nguyên 59, số mũ alpha là 120. Do đó 59 = α120 . Nhân đa thức sinh với α120:

(α120 × α0)x17 + (α120 × α251)x16 + (α120 × α67)x15 + (α120 × α46)x14 + (α120 × α61)x13 + (α120 × α118)x12 + (α120 × α70)x11 + (α120 × α64)x10 + (α120 × α94)x9 + (α120 × α32)x8 + (α120 × α45)x7

Số mũ của các chữ cái được cộng lại với nhau. Trong trường hợp này, ít nhất một trong các số mũ lớn hơn 255, do đó hãy thực hiện module 255 như sau:

α120x17 + α(371 % 255)x16 + α187x15 + α166x14 + α181x13 + α238x12 + α190x11 + α184x10 + α214x9 + α152x8 + α165x7

Kết quả là:

α120x17 + α116x16 + α187x15 + α166x14 + α181x13 + α238x12 + α190x11 + α184x10 + α214x9 + α152x8 + α165x7

Bây giờ, hãy chuyển đổi ký hiệu này thành ký hiệu số nguyên:

59x17 + 248x16 + 220x15 + 63x14 + 49x13 + 11x12 + 174x11 + 149x10 + 249x9 + 73x8 + 145x7

*Bước 9b: XOR kết quả với kết quả ở bước 8b*

Sử dụng kết quả từ bước 8b để thực hiện phép XOR tiếp theo.

(59 ⊕ 59)x17 + (64 ⊕ 248)x16 + (106 ⊕ 220)x15 + (110 ⊕ 63)x14 + (255 ⊕ 49)x13 + (96 ⊕ 11)x12 + (4 ⊕ 174)x11 + (7 ⊕ 149)x10 + (199 ⊕ 249)x9 + (224 ⊕ 73)x8 + (0 ⊕ 145)x7

Kết quả là:

0x17 + 184x16 + 182x15 + 81x14 + 206x13 + 107x12 + 170x11 + 146x10 + 62x9 + 169x8 + 145x7

Loại bỏ số hạng dẫn đầu 0 để có được:

184x16 + 182x15 + 81x14 + 206x13 + 107x12 + 170x11 + 146x10 + 62x9 + 169x8 + 145x7

*Bước 10a: Nhân Đa thức tạo với Số hạng dẫn của kết quả XOR ở bước trước*

Tiếp theo, nhân đa thức tạo với số hạng dẫn đầu của kết quả XOR ở bước trước. Số hạng dẫn đầu trong trường hợp này là 184x16. Chuyển đổi ký hiệu 184x16 sang alpha. Theo bảng log antilog, đối với giá trị nguyên 184, số mũ alpha là 132. Do đó 184 = α132. Nhân đa thức sinh với α132:

(α132 × α0)x16 + (α132 × α251)x15 + (α132 × α67)x14 + (α132 × α46)x13 + (α132 × α61)x12 + (α132 × α118)x11 + (α132 × α70)x10 + (α132 × α64)x9 + (α132 × α94)x8 + (α132 × α32)x7 + (α132 × α45)x6

Số mũ của các chữ cái được cộng lại với nhau. Trong trường hợp này, ít nhất một trong các số mũ lớn hơn 255, do đó hãy thực hiện module 255 như sau:

α132x16 + α(383 % 255)x15 + α199x14 + α178x13 + α193x12 + α250x11 + α202x10 + α196x9 + α226x8 + α164x7 + α177x6

Kết quả là:

α132x16 + α128x15 + α199x14 + α178x13 + α193x12 + α250x11 + α202x10 + α196x9 + α226x8 + α164x7 + α177x6

Bây giờ, hãy chuyển đổi ký hiệu này thành ký hiệu số nguyên:

184x16 + 133x15 + 14x14 + 171x13 + 25x12 + 108x11 + 112x10 + 200x9 + 72x8 + 198x7 + 219x6

*Bước 10b: XOR kết quả với kết quả ở bước 9b*

Sử dụng kết quả từ bước 9b để thực hiện phép XOR tiếp theo.

(184 ⊕ 184)x16 + (182 ⊕ 133)x15 + (81 ⊕ 14)x14 + (206 ⊕ 171)x13 + (107 ⊕ 25)x12 + (170 ⊕ 108)x11 + (146 ⊕ 112)x10 + (62 ⊕ 200)x9 + (169 ⊕ 72)x8 + (145 ⊕ 198)x7 + (0 ⊕ 219)x6

Kết quả là:

0x16 + 51x15 + 95x14 + 101x13 + 114x12 + 198x11 + 226x10 + 246x9 + 225x8 + 87x7 + 219x6

Loại bỏ số hạng dẫn đầu 0 để có được:

51x15 + 95x14 + 101x13 + 114x12 + 198x11 + 226x10 + 246x9 + 225x8 + 87x7 + 219x6

*Bước 11a: Nhân Đa thức tạo với Số hạng dẫn của kết quả XOR ở bước trước*

Tiếp theo, nhân đa thức tạo với số hạng dẫn đầu của kết quả XOR ở bước trước. Số hạng dẫn đầu trong trường hợp này là 51x15. Chuyển đổi ký hiệu 51x15 sang alpha. Theo bảng log antilog, đối với giá trị số nguyên 51, số mũ alpha là 125. Do đó 51 = α125. Nhân đa thức sinh với α125:

(α125 × α0)x15 + (α125 × α251)x14 + (α125 × α67)x13 + (α125 × α46)x12 + (α125 × α61)x11 + (α125 × α118)x10 + (α125 × α70)x9 + (α125 × α64)x8 + (α125 × α94)x7 + (α125 × α32)x6 + (α125 × α45)x5

Số mũ của các chữ cái được cộng lại với nhau. Trong trường hợp này, ít nhất một trong các số mũ lớn hơn 255, do đó hãy thực hiện module 255 như sau:

α125x15 + α(376 % 255)x14 + α192x13 + α171x12 + α186x11 + α243x10 + α195x9 + α189x8 + α219x7 + α157x6 + α170x5

Kết quả là:

α125x15 + α121x14 + α192x13 + α171x12 + α186x11 + α243x10 + α195x9 + α189x8 + α219x7 + α157x6 + α170x5

Bây giờ, hãy chuyển đổi ký hiệu này thành ký hiệu số nguyên:

51x15 + 118x14 + 130x13 + 179x12 + 110x11 + 125x10 + 100x9 + 87x8 + 86x7 + 213x6 + 215x5

*Bước 11b: XOR kết quả với kết quả ở bước 10b*

Sử dụng kết quả từ bước 10b để thực hiện phép XOR tiếp theo.

(51 ⊕ 51)x15 + (95 ⊕ 118)x14 + (101 ⊕ 130)x13 + (114 ⊕ 179)x12 + (198 ⊕ 110)x11 + (226 ⊕ 125)x10 + (246 ⊕ 100)x9 + (225 ⊕ 87)x8 + (87 ⊕ 86)x7 + (219 ⊕ 213)x6 + (0 ⊕ 215)x5

Kết quả là:

0x15 + 41x14 + 231x13 + 193x12 + 168x11 + 159x10 + 146x9 + 182x8 + 1x7 + 14x6 + 215x5

Loại bỏ số hạng dẫn đầu 0 để có được:

41x14 + 231x13 + 193x12 + 168x11 + 159x10 + 146x9 + 182x8 + 1x7 + 14x6 + 215x5

*Bước 12a: Nhân Đa thức tạo với Số hạng dẫn của kết quả XOR ở bước trước*

Tiếp theo, nhân đa thức tạo với số hạng dẫn đầu của kết quả XOR ở bước trước. Số hạng dẫn đầu trong trường hợp này là 41x14. Chuyển đổi ký hiệu 41x14 sang alpha. Theo bảng log antilog, đối với giá trị số nguyên 41, số mũ alpha là 147. Do đó 41 = α147. Nhân đa thức sinh với α147:

(α147 × α0)x14 + (α147 × α251)x13 + (α147 × α67)x12 + (α147 × α46)x11 + (α147 × α61)x10 + (α147 × α118)x9 + (α147 × α70)x8 + (α147 × α64)x7 + (α147 × α94)x6 + (α147 × α32)x5 + (α147 × α45)x4

Số mũ của các chữ cái được cộng lại với nhau. Trong trường hợp này, ít nhất một trong các số mũ lớn hơn 255, do đó hãy thực hiện module 255 như sau:

α147x14 + α(398 % 255)x13 + α214x12 + α193x11 + α208x10 + α(265 % 255)x9 + α217x8 + α211x7 + α24x6 + α179x5 + α192x4

Kết quả là:

α147x14 + α143x13 + α214x12 + α193x11 + α208x10 + α10x9 + α217x8 + α211x7 + α24x6 + α179x5 + α192x4

Bây giờ, hãy chuyển đổi ký hiệu này thành ký hiệu số nguyên:

41x14 + 84x13 + 249x12 + 25x11 + 81x10 + 116x9 + 155x8 + 178x7 + 88x6 + 75x5 + 130x4

*Bước 12b: XOR kết quả với kết quả ở bước 11b*

Sử dụng kết quả từ bước 11b để thực hiện phép XOR tiếp theo.

(41 ⊕ 41)x14 + (231 ⊕ 84)x13 + (193 ⊕ 249)x12 + (168 ⊕ 25)x11 + (159 ⊕ 81)x10 + (146 ⊕ 116)x9 + (182 ⊕ 155)x8 + (1 ⊕ 178)x7 + (14 ⊕ 88)x6 + (215 ⊕ 75)x5 + (0 ⊕ 130)x4

Kết quả là:

0x14 + 179x13 + 56x12 + 177x11 + 206x10 + 230x9 + 45x8 + 179x7 + 86x6 + 156x5 + 130x4

Loại bỏ số hạng dẫn đầu 0 để có được:

179x13 + 56x12 + 177x11 + 206x10 + 230x9 + 45x8 + 179x7 + 86x6 + 156x5 + 130x4

*Bước 13a: Nhân Đa thức tạo với Số hạng dẫn của kết quả XOR ở bước trước*

Tiếp theo, nhân đa thức tạo với số hạng dẫn đầu của kết quả XOR ở bước trước. Số hạng dẫn đầu trong trường hợp này là 179x13 . Chuyển đổi ký hiệu 179x13 sang alpha. Theo bảng log antilog, đối với giá trị số nguyên 179, số mũ alpha là 171. Do đó 179 = α171. Nhân đa thức sinh với α171 :

(α171 × α0)x13 + (α171 × α251)x12 + (α171 × α67)x11 + (α171 × α46)x10 + (α171 × α61)x9 + (α171 × α118)x8 + (α171 × α70)x7 + (α171 × α64)x6 + (α171 × α94)x5 + (α171 × α32)x4 + (α171 × α45)x3

Số mũ của các chữ cái được cộng lại với nhau. Trong trường hợp này, ít nhất một trong các số mũ lớn hơn 255, do đó hãy thực hiện module 255 như sau:

α171x13 + α(422 % 255)x12 + α238x11 + α217x10 + α232x9 + α(289 % 255)x8 + α241x7 + α235x6 + α( 265 % 255)x5 + α203x4 + α216x3

Kết quả là:

α171x13 + α167x12 + α238x11 + α217x10 + α232x9 + α34x8 + α241x7 + α235x6 + α10x5 + α203x4 + α216x3

Bây giờ, hãy chuyển đổi ký hiệu này thành ký hiệu số nguyên:

179x13 + 126x12 + 11x11 + 155x10 + 247x9 + 78x8 + 88x7 + 235x6 + 116x5 + 224x4 + 195x3

*Bước 13b: XOR kết quả với kết quả ở bước 12b*

Sử dụng kết quả từ bước 12b để thực hiện phép XOR tiếp theo.

(179 ⊕ 179)x13 + (56 ⊕ 126)x12 + (177 ⊕ 11)x11 + (206 ⊕ 155)x10 + (230 ⊕ 247)x9 + (45 ⊕ 78)x8 + (179 ⊕ 88)x7 + (86 ⊕ 235)x6 + (156 ⊕ 116)x5 + (130 ⊕ 224)x4 + (0 ⊕ 195)x3

Kết quả là:

0x13 + 70x12 + 186x11 + 85x10 + 17x9 + 99x8 + 235x7 + 189x6 + 232x5 + 98x4 + 195x3

Loại bỏ số hạng dẫn đầu 0 để có được:

70x12 + 186x11 + 85x10 + 17x9 + 99x8 + 235x7 + 189x6 + 232x5 + 98x4 + 195x3

*Bước 14a: Nhân Đa thức tạo với Số hạng dẫn của kết quả XOR ở bước trước*

Tiếp theo, nhân đa thức tạo với số hạng dẫn đầu của kết quả XOR ở bước trước. Số hạng dẫn đầu trong trường hợp này là 70x12 . Chuyển đổi ký hiệu 70x12 sang alpha. Theo bảng log antilog, đối với giá trị số nguyên 70, số mũ alpha là 48. Do đó 70 = α48. Nhân đa thức sinh với α48:

(α48 × α0)x12 + (α48 × α251)x11 + (α48 × α67)x10 + (α48 × α46)x9 + (α48 × α61)x8 + (α48 × α118)x7 + (α48 × α70)x6 + (α48 × α64)x5 + (α48 × α94)x4 + (α48 × α32)x3 + (α48 × α45)x2

Số mũ của các chữ cái được cộng lại với nhau. Trong trường hợp này, ít nhất một trong các số mũ lớn hơn 255, do đó hãy thực hiện module 255 như sau:

α48x12 + α(299 % 255)x11 + α115x10 + α94x9 + α109x8 + α166x7 + α118x6 + α112x5 + α142x4 + α80x3 + α93x2

Kết quả là:

α48x12 + α44x11 + α115x10 + α94x9 + α109x8 + α166x7 + α118x6 + α112x5 + α142x4 + α80x3 + α93x2

Bây giờ, hãy chuyển đổi ký hiệu này thành ký hiệu số nguyên:

70x12 + 238x11 + 124x10 + 113x9 + 189x8 + 63x7 + 199x6 + 129x5 + 42x4 + 253x3 + 182x2

*Bước 14b: XOR kết quả với kết quả ở bước 13b*

Sử dụng kết quả từ bước 13b để thực hiện phép XOR tiếp theo.

(70 ⊕ 70)x12 + (186 ⊕ 238)x11 + (85 ⊕ 124)x10 + (17 ⊕ 113)x9 + (99 ⊕ 189)x8 + (235 ⊕ 63)x7 + (189 ⊕ 199)x6 + (232 ⊕ 129)x5 + (98 ⊕ 42)x4 + (195 ⊕ 253)x3 + (0 ⊕ 182)x2

Kết quả là:

0x12 + 84x11 + 41x10 + 96x9 + 222x8 + 212x7 + 122x6 + 105x5 + 72x4 + 62x3 + 182x2

Loại bỏ số hạng dẫn đầu 0 để có được:

84x11 + 41x10 + 96x9 + 222x8 + 212x7 + 122x6 + 105x5 + 72x4 + 62x3 + 182x2

*Bước 15a: Nhân Đa thức tạo với Số hạng dẫn của kết quả XOR ở bước trước*

Tiếp theo, nhân đa thức tạo với số hạng dẫn đầu của kết quả XOR ở bước trước. Số hạng dẫn đầu trong trường hợp này là 84x11. Chuyển đổi ký hiệu 84x11 sang alpha. Theo bảng log antilog, đối với giá trị số nguyên 84, số mũ alpha là 143. Do đó 84 = α143. Nhân đa thức sinh với α143:

(α143 × α0)x11 + (α143 × α251)x10 + (α143 × α67)x9 + (α143 × α46)x8 + (α143 × α61)x7 + (α143 × α118)x6 + (α143 × α70)x5 + (α143 × α64)x4 + (α143 × α94)x3 + (α143 × α32)x2 + (α143 × α45)x1

Số mũ của các chữ cái được cộng lại với nhau. Trong trường hợp này, ít nhất một trong các số mũ lớn hơn 255, do đó hãy thực hiện module 255 như sau:

α143x11 + α(394 % 255)x10 + α210x9 + α189x8 + α204x7 + α(261 % 255)x6 + α213x5 + α207x4 + α237x3 + α175x2 + α188x1

Kết quả là:

α143x11 + α139x10 + α210x9 + α189x8 + α204x7 + α6x6 + α213x5 + α207x4 + α237x3 + α175x2 + α188x1

Bây giờ, hãy chuyển đổi ký hiệu này thành ký hiệu số nguyên:

84x11 + 66x10 + 89x9 + 87x8 + 221x7 + 64x6 + 242x5 + 166x4 + 139x3 + 255x2 + 165x1

*Bước 15b: XOR kết quả với kết quả ở bước 14b*

Sử dụng kết quả từ bước 14b để thực hiện phép XOR tiếp theo.

(84 ⊕ 84)x11 + (41 ⊕ 66)x10 + (96 ⊕ 89)x9 + (222 ⊕ 87)x8 + (212 ⊕ 221)x7 + (122 ⊕ 64)x6 + (105 ⊕ 242)x5 + (72 ⊕ 166)x4 + (62 ⊕ 139)x3 + (182 ⊕ 255)x2 + (0 ⊕ 165)x1

Kết quả là:

0x11 + 107x10 + 57x9 + 137x8 + 9x7 + 58x6 + 155x5 + 238x4 + 181x3 + 73x2 + 165x1

Loại bỏ số hạng dẫn đầu 0 để có được:

107x10 + 57x9 + 137x8 + 9x7 + 58x6 + 155x5 + 238x4 + 181x3 + 73x2 + 165x1

*Bước 16a: Nhân Đa thức tạo với Số hạng dẫn của kết quả XOR ở bước trước*

Tiếp theo, nhân đa thức tạo với số hạng dẫn đầu của kết quả XOR ở bước trước. Số hạng dẫn đầu trong trường hợp này là 107x10. Chuyển đổi ký hiệu 107x10 sang alpha. Theo bảng log antilog, đối với giá trị số nguyên 107, số mũ alpha là 84. Do đó 107 = α84. Nhân đa thức sinh với α84:

(α84 × α0)x10 + (α84 × α251)x9 + (α84 × α67)x8 + (α84 × α46)x7 + (α84 × α61)x6 + (α84 × α118)x5 + (α84 × α70)x4 + (α84 × α64)x3 + (α84 × α94)x2 + (α84 × α32)x1 + (α84 × α45)x0

Số mũ của các chữ cái được cộng lại với nhau. Trong trường hợp này, ít nhất một trong các số mũ lớn hơn 255, do đó hãy thực hiện module 255 như sau:

α84x10 + α(335 % 255)x9 + α151x8 + α130x7 + α145x6 + α202x5 + α154x4 + α148x3 + α178x2 + α116x1 + α129x0

Kết quả là:

α84x10 + α80x9 + α151x8 + α130x7 + α145x6 + α202x5 + α154x4 + α148x3 + α178x2 + α116x1 + α129x0

Bây giờ, hãy chuyển đổi ký hiệu này thành ký hiệu số nguyên:

107x10 + 253x9 + 170x8 + 46x7 + 77x6 + 112x5 + 57x4 + 82x3 + 171x2 + 248x1 + 23

*Bước 16b: XOR kết quả với kết quả ở bước 15b*

Sử dụng kết quả từ bước 15b để thực hiện phép XOR tiếp theo.

(107 ⊕ 107)x10 + (57 ⊕ 253)x9 + (137 ⊕ 170)x8 + (9 ⊕ 46)x7 + (58 ⊕ 77)x6 + (155 ⊕ 112)x5 + (238 ⊕ 57)x4 + (181 ⊕ 82)x3 + (73 ⊕ 171)x2 + (165 ⊕ 248)x1 + (0 ⊕ 23)x0

Kết quả là:

0x10 + 196x9 + 35x8 + 39x7 + 119x6 + 235x5 + 215x4 + 231x3 + 226x2 + 93x1 + 23

Loại bỏ số hạng dẫn đầu 0 để có được:

196x9 + 35x8 + 39x7 + 119x6 + 235x5 + 215x4 + 231x3 + 226x2 + 93x1 + 23

*Sử dụng các số hạng còn lại làm từ mã sửa lỗi*

Phép chia đã được thực hiện 16 lần, đó là số số hạng trong đa thức thông điệp. Điều này có nghĩa là phép chia đã hoàn tất và các số hạng của đa thức trên là các từ mã sửa lỗi được sử dụng cho đa thức thông báo ban đầu:

196 35 39 119 235 215 231 226 93 23

**Cấu trúc thông báo cuối cùng**

Trong phần mã hóa sửa lỗi, quy trình sửa lỗi Reed-Solomon đã được giải thích. Bây giờ bạn sẽ có các từ mã dữ liệu và các từ mã sửa lỗi tương ứng của chúng. Như đã đề cập ở bước trước, mã QR lớn hơn yêu cầu bạn chia từ mã dữ liệu thành các khối nhỏ hơn và tạo từ mã sửa lỗi riêng cho từng khối. Trong trường hợp này, các khối dữ liệu và từ mã sửa lỗi phải được xen kẽ theo đặc tả mã QR. Phần này giải thích chi tiết quá trình xen kẽ.

*Bước 1: Xác định cần bao nhiêu khối và từ mã sửa lỗi*

Bảng sửa lỗi cho biết số lượng khối dữ liệu và từ mã sửa lỗi trên mỗi khối được yêu cầu cho mỗi phiên bản và mức sửa lỗi.

*Đối với các mã nhỏ hơn, hãy sử dụng các từ mã sửa lỗi và dữ liệu nguyên trạng*

Xin lưu ý rằng các mã QR nhỏ hơn chỉ bao gồm một khối từ mã dữ liệu, với một bộ từ mã sửa lỗi cho khối đó. Trong trường hợp này, không cần xen kẽ. Đơn giản chỉ cần đặt các từ mã sửa lỗi sau các từ mã dữ liệu và chuyển sang bước tiếp theo, đặt module vào ma trận.

*Chia nhỏ mã QR lớn hơn*

Trên trang mã hóa sửa lỗi, tôi đã đưa ra một ví dụ sử dụng mã 5-Q. Bảng sửa lỗi cho biết nhóm đầu tiên của mã 5-Q bao gồm 2 khối, với 15 từ mã dữ liệu trên mỗi khối và nhóm thứ hai bao gồm 2 khối, với 16 từ mã dữ liệu trên mỗi khối. Đây lại là bảng hiển thị trên trang mã hóa sửa lỗi cho thấy mã 5-Q được chia thành các nhóm và khối.

|  |  |  |
| --- | --- | --- |
| Số nhóm | Chặn số | Từ mã dữ liệu trong nhóm |
| Nhóm 1 | Khối 1 | (codeword #1) 01000011  (codeword #2) 01010101  (codeword #3) 01000110  (codeword #4) 10000110  (codeword #5) 01010111  (codeword #6) 00100110  (codeword #7) 01010101  (codeword #8) 11000010  (codeword #9) 01110111  (codeword #10) 00110010  (codeword #11) 00000110  (codeword #12) 00010010  (codeword #13) 00000110  (codeword #14) 01100111  (codeword #15) 00100110 |
| Khối 2 | (codeword #16) 11110110  (codeword #17) 11110110  (codeword #18) 01000010  (codeword #19) 00000111  (codeword #20) 01110110  (codeword #21) 10000110  (codeword #22) 11110010  (codeword #23) 00000111  (codeword #24) 00100110  (codeword #25) 01010110  (codeword #26) 00010110  (codeword #27) 11000110  (codeword #28) 11000111  (codeword #29) 10010010  (codeword #30) 00000110 |
| Nhóm 2 | Khối 1 | (codeword #31) 10110110  (codeword #32) 11100110  (codeword #33) 11110111  (codeword #34) 01110111  (codeword #35) 00110010  (codeword #36) 00000111  (codeword #37) 01110110  (codeword #38) 10000110  (codeword #39) 01010111  (codeword #40) 00100110  (codeword #41) 01010010  (codeword #42) 00000110  (codeword #43) 10000110  (codeword #44) 10010111  (codeword #45) 00110010  (codeword #46) 00000111 |
| Khối 2 | (codeword #47) 01000110  (codeword #48) 11110111  (codeword #49) 01110110  (codeword #50) 01010110  (codeword #51) 11000010  (codeword #52) 00000110  (codeword #53) 10010111  (codeword #54) 00110010  (codeword #55) 11100000  (codeword #56) 11101100  (codeword #57) 00010001  (codeword #58) 11101100  (codeword #59) 00010001  (codeword #60) 11101100  (codeword #61) 00010001  (codeword #62) 11101100 |

Lưu ý rằng bảng sửa lỗi cho biết rằng với mỗi khối, mã 5-Q phải có 18 từ mã sửa lỗi. Bây giờ mã sửa lỗi đã được giải thích, các từ mã sửa lỗi cho dữ liệu trên có thể được tính toán.

Có bốn khối, do đó cần tạo ra bốn bộ từ mã sửa lỗi 18. Sau đây là bảng cập nhật hiển thị các từ mã sửa dữ liệu của từng khối được chuyển đổi thành danh sách các số nguyên và 18 từ mã sửa lỗi được tạo cho mỗi khối.

|  |  |  |  |
| --- | --- | --- | --- |
| Số nhóm | Chặn số | Từ mã dữ liệu trong nhóm | Từ mã dữ liệu dưới dạng số nguyên |
| Từ mã sửa lỗi cho khối |
| Nhóm 1 | Khối 1 | (codeword #1) 01000011  (codeword #2) 01010101  (codeword #3) 01000110  (codeword #4) 10000110  (codeword #5) 01010111  (codeword #6) 00100110  (codeword #7) 01010101  (codeword #8) 11000010  (codeword #9) 01110111  (codeword #10) 00110010  (codeword #11) 00000110  (codeword #12) 00010010  (codeword #13) 00000110  (codeword #14) 01100111  (codeword #15) 00100110 | Dữ liệu Từ mã từ bên trái, dưới dạng Số nguyên: 67,85,70,134,87,38,85,194,119,50,6,18,6,103,38  Sửa lỗi Từ mã (hiển thị cách tính toán ):  213 199 11 45 115 247 241 223 229 248 154 117 154 111 86 161 111 39 |
| Khối 2 | (codeword #16) 11110110  (codeword #17) 11110110  (codeword #18) 01000010  (codeword #19) 00000111  (codeword #20) 01110110  (codeword #21) 10000110  (codeword #22) 11110010  (codeword #23) 00000111  (codeword #24) 00100110  (codeword #25) 01010110  (codeword #26) 00010110  (codeword #27) 11000110  (codeword #28) 11000111  (codeword #29) 10010010  (codeword #30) 00000110 | Dữ liệu Từ mã từ bên trái, dưới dạng Số nguyên:  246,246,66,7,118,134,242,7,38,86,22,198,199,146,6  Sửa lỗi Từ mã (hiển thị cách tính toán ):  87 204 96 60 202 182 124 157 200 134 27 129 209 17 163 163 120 133 |
| Nhóm 2 | Khối 1 | (codeword #31) 10110110  (codeword #32) 11100110  (codeword #33) 11110111  (codeword #34) 01110111  (codeword #35) 00110010  (codeword #36) 00000111  (codeword #37) 01110110  (codeword #38) 10000110  (codeword #39) 01010111  (codeword #40) 00100110  (codeword #41) 01010010  (codeword #42) 00000110  (codeword #43) 10000110  (codeword #44) 10010111  (codeword #45) 00110010  (codeword #46) 00000111 | Từ mã dữ liệu từ bên trái, dưới dạng Số nguyên:  182,230,247,119,50,7,118,134,87,38,82,6,134,151,50,7  Từ mã sửa lỗi (hiển thị cách tính toán ):  148 116 177 212 76 133 75 242 238 76 195 230 189 10 108 240 192 141 |
| Khối 2 | (codeword #47) 01000110  (codeword #48) 11110111  (codeword #49) 01110110  (codeword #50) 01010110  (codeword #51) 11000010  (codeword #52) 00000110  (codeword #53) 10010111  (codeword #54) 00110010  (codeword #55) 11100000  (codeword #56) 11101100  (codeword #57) 00010001  (codeword #58) 11101100  (codeword #59) 00010001  (codeword #60) 11101100  (codeword #61) 00010001  (codeword #62) 11101100 | Từ mã dữ liệu từ bên trái, dưới dạng Số nguyên:  70,247,118,86,194,6,151,50,16,236,17,236,17,236,17,236  Từ mã sửa lỗi (hiển thị cách tính toán ):  235 159 5 173 24 147 59 33 106 40 255 172 82 2 131 32 178 236 |

Bây giờ các từ mã sửa lỗi cho dữ liệu đã được tính toán, phần tiếp theo sẽ giải thích quá trình xen kẽ các khối dữ liệu và các từ mã sửa lỗi.

*Bước 2: Xen kẽ các khối*

Các khối được xen kẽ bằng cách thực hiện như sau:

1. lấy từ mã dữ liệu đầu tiên từ khối đầu tiên
2. theo sau là từ mã dữ liệu đầu tiên từ khối thứ hai
3. theo sau là từ mã dữ liệu đầu tiên từ khối thứ ba
4. theo sau là từ mã dữ liệu đầu tiên từ khối thứ tư
5. theo sau là từ mã dữ liệu thứ hai từ khối đầu tiên
6. và như thế

Mẫu này được lặp lại, đi qua các khối, cho đến khi tất cả các từ mã dữ liệu được xen kẽ.

Sau đó, hãy làm như sau:

1. lấy từ mã sửa lỗi đầu tiên từ khối đầu tiên
2. theo sau là từ mã sửa lỗi đầu tiên từ khối thứ hai
3. theo sau là từ mã sửa lỗi đầu tiên từ khối thứ ba
4. theo sau là từ mã sửa lỗi đầu tiên từ khối thứ tư
5. theo sau là từ mã sửa lỗi thứ hai từ khối đầu tiên
6. và như thế

Thực hiện việc này cho đến khi tất cả các từ mã sửa lỗi đã được sử dụng hết.

Quá trình này được trình bày chi tiết dưới đây.

*Xen kẽ các từ mã dữ liệu*

Quá trình xen kẽ các từ mã dữ liệu được minh họa bằng bảng và mô tả bên dưới.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Khối 1 | Khối 2 | Khối 3 | Khối 4 |
| Col 1 | 67 | 246 | 182 | 70 |
| Col 2 | 85 | 246 | 230 | 247 |
| Col 3 | 70 | 66 | 247 | 118 |
| Col 4 | 134 | 7 | 119 | 86 |
| Col 5 | 87 | 118 | 50 | 194 |
| Col 6 | 38 | 134 | 7 | 6 |
| Col 7 | 85 | 242 | 118 | 151 |
| Col 8 | 194 | 7 | 134 | 50 |
| Col 9 | 119 | 38 | 87 | 16 |
| Col 10 | 50 | 86 | 38 | 236 |
| Col 11 | 6 | 22 | 82 | 17 |
| Col 12 | 18 | 198 | 6 | 236 |
| Col 13 | 6 | 199 | 134 | 17 |
| Col 14 | 103 | 146 | 151 | 236 |
| Col 15 | 38 | 6 | 50 | 17 |
| Col 16 |  |  | 7 | 236 |

Như mô tả ở trên, lấy từ mã dữ liệu đầu tiên từ khối thứ nhất, tiếp theo là từ mã dữ liệu đầu tiên từ khối thứ hai, tiếp theo là từ mã dữ liệu đầu tiên từ khối thứ ba, tiếp theo là từ mã dữ liệu đầu tiên từ khối thứ tư.

Hoặc, như trong bảng trên, lấy từ mã dữ liệu từ cột 1, bắt đầu bằng khối 1 và kết thúc ở khối 4. Các số từ cột 1 là 67, 246, 182 và 70.

Dữ liệu xen kẽ cho đến nay:

67, 246, 182, 70

Sau đó, điền các số ở cột 2, bắt đầu từ khối 1 và kết thúc ở khối 4.

Dữ liệu xen kẽ cho đến nay:

67, 246, 182, 70, 85, 246, 230, 247

Làm tương tự với cột 3.

Dữ liệu xen kẽ cho đến nay:

67, 246, 182, 70, 85, 246, 230, 247, 70, 66, 247, 118

Lặp lại quá trình, đi từng cột. Đây là kết quả sau khi thêm cột 15.

Dữ liệu xen kẽ sau khi thêm cột 15:

67, 246, 182, 70, 85, 246, 230, 247, 70, 66, 247, 118, 134, 7, 119, 86, 87, 118, 50, 194, 38, 134 , 7, 6, 85, 242, 118, 151, 194, 7, 134, 50, 119, 38, 87, 16, 50, 86, 38, 236, 6, 22, 82, 17, 18, 198, 6 , 236, 6, 199, 134, 17, 103, 146, 151, 236, 38, 6, 50, 17

Cột 16 chỉ có hai số trong đó, như bạn có thể thấy trong bảng trên. Chỉ cần đặt hai số này vào cuối dữ liệu xen kẽ như sau:

Các từ mã dữ liệu xen kẽ:

67, 246, 182, 70, 85, 246, 230, 247, 70, 66, 247, 118, 134, 7, 119, 86, 87, 118, 50, 194, 38, 134, 7, 6, 85, 242, 118, 151, 194, 7, 134, 50, 119, 38, 87, 16, 50, 86, 38, 236, 6, 22, 82, 17, 18, 198, 6, 236, 6, 199, 134, 17, 103, 146, 151, 236, 38, 6, 50, 17, 7, 236.

*Xen kẽ các từ mã sửa lỗi*

Bây giờ các từ mã dữ liệu đã được xen kẽ, các từ mã sửa lỗi sẽ được xen kẽ. Quá trình này được minh họa bằng bảng và mô tả dưới đây.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Khối 1 | Khối 2 | Khối 3 | Khối 4 |
| Col 1 | 213 | 87 | 148 | 235 |
| Col 2 | 199 | 204 | 116 | 159 |
| Col 3 | 11 | 96 | 177 | 5 |
| Col 4 | 45 | 60 | 212 | 173 |
| Col 5 | 115 | 202 | 76 | 24 |
| Col 6 | 247 | 182 | 133 | 147 |
| Col 7 | 241 | 124 | 75 | 59 |
| Col 8 | 223 | 157 | 242 | 33 |
| Col 9 | 229 | 200 | 238 | 106 |
| Col 10 | 248 | 134 | 76 | 40 |
| Col 11 | 154 | 27 | 195 | 255 |
| Col 12 | 117 | 129 | 230 | 172 |
| Col 13 | 154 | 209 | 189 | 82 |
| Col 14 | 111 | 17 | 10 | 2 |
| Col 15 | 86 | 163 | 108 | 131 |
| Col 16 | 161 | 163 | 240 | 32 |
| Col 17 | 111 | 120 | 192 | 178 |
| Col 18 | 39 | 133 | 141 | 236 |

Như mô tả ở trên, lấy từ mã sửa lỗi đầu tiên từ khối đầu tiên, tiếp theo là từ mã sửa lỗi đầu tiên từ khối thứ hai, tiếp theo là từ mã sửa lỗi đầu tiên từ khối thứ ba, tiếp theo là từ mã sửa lỗi đầu tiên từ khối thứ tư. . Hoặc, như thể hiện trong bảng trên, lấy từ mã sửa lỗi từ cột 1, bắt đầu từ khối 1 và kết thúc ở khối 4. Các số từ cột 1 là 213, 87, 148 và 235.

Các từ mã sửa lỗi xen kẽ cho đến nay:

213, 87, 148, 235

Sau đó, điền các số ở cột 2, bắt đầu từ khối 1 và kết thúc ở khối 4.

Các từ mã sửa lỗi xen kẽ cho đến nay:

213, 87, 148, 235, 199, 204, 116, 159

Làm tương tự với cột 3.

Các từ mã sửa lỗi xen kẽ cho đến nay:

213, 87, 148, 235, 199, 204, 116, 159, 11, 96, 177, 5

Lặp lại quá trình, đi từng cột. Đây là kết quả cuối cùng.

Các từ mã sửa lỗi xen kẽ:

213, 87, 148, 235, 199, 204, 116, 159, 11, 96, 177, 5, 45, 60, 212, 173, 115, 202, 76, 24, 247, 182, 133 , 147, 241, 124, 75, 59, 223, 157, 242, 33, 229, 200, 238, 106, 248, 134, 76, 40, 154, 27, 195, 255, 117, 129, 230, 172 , 154, 209, 189, 82, 111, 17, 10, 2, 86, 163, 108, 131, 161, 163, 240, 32, 111, 120, 192, 178, 39, 133, 141, 236

*Đặt các từ mã sửa lỗi xen kẽ sau các từ mã dữ liệu xen kẽ*

Thông báo cuối cùng bao gồm các từ mã dữ liệu được xen kẽ, theo sau là các từ mã sửa lỗi được xen kẽ.

Tin nhắn cuối cùng:

67, 246, 182, 70, 85, 246, 230, 247, 70, 66, 247, 118, 134, 7, 119, 86, 87, 118, 50, 194, 38, 134, 7, 6 , 85, 242, 118, 151, 194, 7, 134, 50, 119, 38, 87, 16, 50, 86, 38, 236, 6, 22, 82, 17, 18, 198, 6, 236, 6 , 199, 134, 17, 103, 146, 151, 236, 38, 6, 50, 17, 7, 236, 213, 87, 148, 235, 199, 204, 116, 159, 11, 96, 177, 5 , 45, 60, 212, 173, 115, 202, 76, 24, 247, 182, 133, 147, 241, 124, 75, 59, 223, 157, 242, 33, 229, 200, 238, 106, 248 , 134, 76, 40, 154, 27, 195, 255, 117, 129, 230, 172, 154, 209, 189, 82, 111, 17, 10, 2, 86, 163, 108, 131, 161, 163 , 240, 32, 111, 120, 192, 178, 39, 133, 141, 236

*Bước 3: Chuyển đổi sang nhị phân*

Thông báo cuối cùng được chuyển đổi sang dạng nhị phân 8 bit. Trong ví dụ của chúng tôi, bốn từ mã đầu tiên là 67, 246, 182 và 70. Được chuyển đổi sang nhị phân, đó là:

67 = 01000011

246 = 11110110

182 = 10110110

70 = 01000110

Các số nhị phân 8 bit được ghép lại với nhau thành một chuỗi liền mạch. Sử dụng bốn từ mã đầu tiên ở trên, 32 bit đầu tiên của thông báo cuối cùng là:

01000011111101101011011001000110

Để hoàn thiện, toàn bộ thông báo cuối cùng ở dạng nhị phân nằm trong hộp văn bản bên dưới:

0100001111110110101101100100011001010101111101101110011011110111010001100100001011110111011101101000011000000111011101110101011001010111011101100011001011000010001001101000011000000111000001100101010111110010011101101001011111000010000001111000011000110010011101110010011001010111000100000011001001010110001001101110110000000110000101100101001000010001000100101100011000000110111011000000011011000111100001100001000101100111100100101001011111101100001001100000011000110010000100010000011111101100110101010101011110010100111010111100011111001100011101001001111100001011011000001011000100000101001011010011110011010100101011010111001111001010010011000001100011110111101101101000010110010011111100010111110001001011001110111101111110011101111100100010000111100101110010001110111001101010111110001000011001001100001010001001101000011011110000111111111101110101100000011110011010101100100110101101000110111101010100100110111100010001000010100000001001010110101000110110110010000011101000011010001111110000001000000110111101111000110000001011001000100111100001011000110111101100

*Bước 4: Thêm số bit còn lại nếu cần thiết*

Đối với một số phiên bản QR, thông báo nhị phân cuối cùng không đủ dài để điền vào số bit cần thiết. Trong trường hợp này, cần thêm một số số 0 nhất định vào cuối tin nhắn cuối cùng để nó có độ dài chính xác. Những số 0 bổ sung này được gọi là các bit còn lại . Mã QR phiên bản 5, giống như mã trong ví dụ này, phải có 7 bit còn lại được thêm vào cuối.

8 bit cuối cùng của chuỗi trong hộp văn bản ở trên là:

11101100

Với 7 bit còn lại được thêm vào cuối, các bit cuối cùng của chuỗi là:

111011000000000

Để hoàn thiện, toàn bộ thông báo cuối cùng ở dạng nhị phân, với 7 bit còn lại được thêm vào, nằm trong hộp văn bản bên dưới:

01000011111101101011011001000110010101011111011011100110111101110100011001000010111101110111011010000110000001110111011101010110010101110111011000110010110000100010011010000110000001110000011001010101111100100111011010010111110000100000011110000110001100100111011100100110010101110001000000110010010101100010011011101100000001100001011001010010000100010001001011000110000001101110110000000110110001111000011000010001011001111001001010010111111011000010011000000110001100100001000100000111111011001101010101010111100101001110101111000111110011000111010010011111000010110110000010110001000001010010110100111100110101001010110101110011110010100100110000011000111101111011011010000101100100111111000101111100010010110011101111011111100111011111001000100001111001011100100011101110011010101111100010000110010011000010100010011010000110111100001111111111011101011000000111100110101011001001101011010001101111010101001001101111000100010000101000000010010101101010001101101100100000111010000110100011111100000010000001101111011110001100000010110010001001111000010110001101111011000000000

Xem bên dưới để biết bảng liệt kê số bit còn lại cần thiết cho mỗi phiên bản QR.

Bây giờ chuỗi nhị phân cuối cùng đã được xác định, bước tiếp theo là đặt các bit dữ liệu và mẫu hàm vào ma trận mã QR. Quá trình này được giải thích trong phần vị trí module trong phần ma trận.

Danh sách các phiên bản và số bit còn lại bắt buộc

Bảng sau liệt kê 40 phiên bản QR và số bit còn lại phải được thêm vào cuối chuỗi thông báo cuối cùng. Lưu ý rằng các bit còn lại được chỉ định cho mỗi phiên bản là bắt buộc cho dù sử dụng mức sửa lỗi nào. Cũng lưu ý rằng một số phiên bản, chẳng hạn như phiên bản 7 đến 13, không yêu cầu thêm bất kỳ bit còn lại nào.

|  |  |
| --- | --- |
| **Phiên bản QR** | **Số bit còn lại bắt buộc** |
| 1 | 0 |
| 2 | 7 |
| 3 | 7 |
| 4 | 7 |
| 5 | 7 |
| 6 | 7 |
| 7 | 0 |
| 8 | 0 |
| 9 | 0 |
| 10 | 0 |
| 11 | 0 |
| 12 | 0 |
| 13 | 0 |
| 14 | 3 |
| 15 | 3 |
| 16 | 3 |
| 17 | 3 |
| 18 | 3 |
| 19 | 3 |
| 20 | 3 |
| 21 | 4 |
| 22 | 4 |
| 23 | 4 |
| 24 | 4 |
| 25 | 4 |
| 26 | 4 |
| 27 | 4 |
| 28 | 3 |
| 29 | 3 |
| 30 | 3 |
| 31 | 3 |
| 32 | 3 |
| 33 | 3 |
| 34 | 3 |
| 35 | 0 |
| 36 | 0 |
| 37 | 0 |
| 38 | 0 |
| 39 | 0 |
| 40 | 0 |

**Vị trí module trong ma trận**

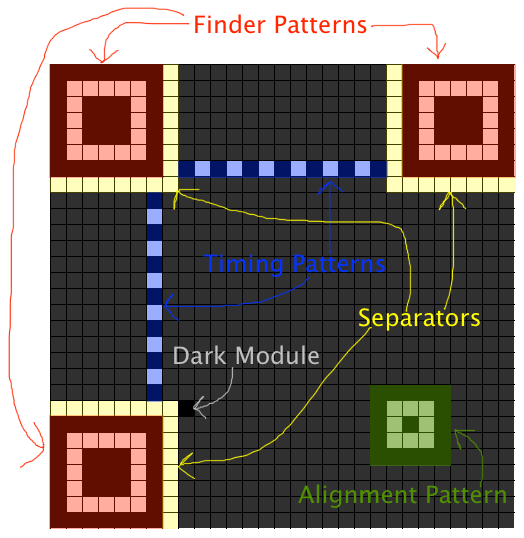
Ở bước trước (cấu trúc thông báo cuối cùng), các từ mã sửa lỗi và dữ liệu được xen kẽ và thu được chuỗi bit cuối cùng. Bước tiếp theo là đưa chúng vào ma trận mã QR cùng với các mẫu chức năng được yêu cầu. Mẫu hàm là thành phần phi dữ liệu của mã QR được đặc tả mã QR yêu cầu, chẳng hạn như ba mẫu công cụ tìm ở các góc của ma trận mã QR. Phần này giải thích thời điểm và cách đặt các mẫu chức năng và bit dữ liệu.

*Thuật ngữ: Pixel so với Module*

Trên trang này, tôi coi các ô vuông đen trắng của mã QR là module chứ không phải pixel. Điều này nhằm phân biệt giữa các pixel trên màn hình và các ô vuông đen trắng của mã QR. Ví dụ: mã QR phiên bản 1 luôn có 21 module x 21 module, ngay cả khi nó chiếm 42 x 42 pixel trên màn hình máy tính hoặc 105x105, v.v.

*Tổng quan về các mẫu hàm*

Mã QR phải bao gồm các mẫu chức năng . Đây là những hình dạng phải được đặt ở những vùng cụ thể của mã QR để đảm bảo máy quét mã QR có thể nhận dạng và định hướng chính xác mã để giải mã. Hình ảnh sau đây đưa ra ví dụ về các mẫu hàm là gì và vị trí của chúng.



Các **mẫu công cụ tìm** là ba khối ở các góc của mã QR ở trên cùng bên trái, trên cùng bên phải và dưới cùng bên trái.

Dấu **phân cách** là các vùng khoảng trắng bên cạnh các mẫu tìm kiếm.

Các **mẫu căn chỉnh** tương tự như các mẫu của công cụ tìm kiếm, nhưng nhỏ hơn và được đặt xuyên suốt mã. Chúng được sử dụng trong phiên bản 2 trở lên và vị trí của chúng phụ thuộc vào phiên bản mã QR.

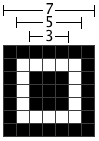
Các **mẫu thời gian** là các đường chấm chấm kết nối các mẫu công cụ tìm.

**Module tối** là một module màu đen duy nhất luôn được đặt bên cạnh mẫu công cụ tìm phía dưới bên trái.

Các phần bên dưới giải thích chi tiết hơn về cách định vị các mẫu chức năng.

*Bước 1: Thêm mẫu Finder*

Đầu tiên, đặt các mẫu tìm kiếm vào ma trận. Mẫu công cụ tìm (hiển thị bên dưới) bao gồm một hình vuông màu đen bên ngoài có 7 module x 7 module, một hình vuông màu trắng bên trong có 5 module x 5 module và một hình vuông màu đen đặc ở giữa có 3 module x 3 module.



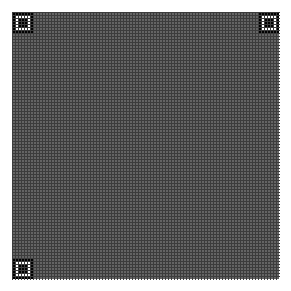
Mẫu công cụ tìm được thiết kế thành mẫu không thể xuất hiện trong các phần khác của mã QR. Độ rộng module của mẫu công cụ tìm có tỷ lệ 1:1:3:1:1. Máy quét mã QR có thể tìm kiếm tỷ lệ module sáng và tối này để phát hiện các mẫu công cụ tìm và định hướng chính xác mã QR để giải mã.

Các mẫu công cụ tìm kiếm **luôn** được đặt ở góc **trên cùng bên trái, trên cùng bên phải và dưới cùng bên trái** của mã QR, bất kể phiên bản nào đang được sử dụng.

Để minh họa điều này, các hình ảnh sau đây hiển thị vị trí của các mẫu công cụ tìm. Hình ảnh đầu tiên là mã phiên bản 1 và hình ảnh thứ hai là mã phiên bản 18.



Phiên bản 1



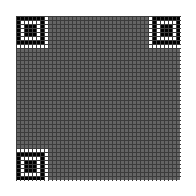
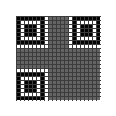
Phiên bản 18

Kích thước của mã QR có thể được tính bằng công thức (((V-1)×4)+21), trong đó V là phiên bản mã QR. Ví dụ: phiên bản 32 là (((32-1)×4)+21) hoặc 145 module x 145 module. Do đó, vị trí của các mẫu tìm kiếm có thể được khái quát như sau:

* Góc trên cùng bên trái của mẫu công cụ tìm phía trên bên trái luôn được đặt ở (0,0).
* Góc trên TRÁI của mẫu tìm kiếm trên cùng bên phải luôn được đặt tại ([(((V-1)×4)+21) - 7], 0)
* Góc TRÁI trên cùng của mẫu công cụ tìm phía dưới bên trái luôn được đặt ở (0,[(((V-1)×4)+21) - 7])

*Bước 2: Thêm dấu phân cách*

Dấu phân cách là các dòng module màu trắng, rộng một module, được đặt bên cạnh các mẫu công cụ tìm để tách chúng khỏi phần còn lại của mã QR. Các dấu phân cách chỉ được đặt bên cạnh các cạnh của mẫu công cụ tìm chạm vào bên trong mã QR. Ví dụ:

****

*Bước 3: Thêm các mẫu căn chỉnh*

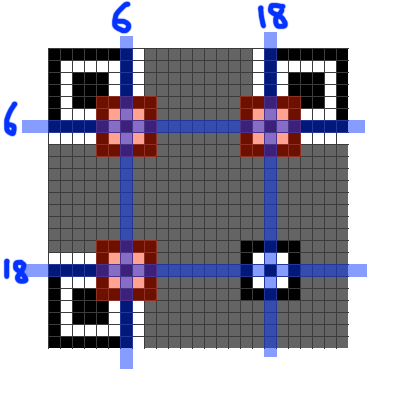
Mã QR phiên bản 2 trở lên bắt buộc phải có mẫu căn chỉnh. Mẫu căn chỉnh, được hiển thị bên dưới, bao gồm một module 5 x 5 hình vuông màu đen module, một module 3 bên trong hình vuông màu trắng module 3 và một module màu đen duy nhất ở giữa.

****

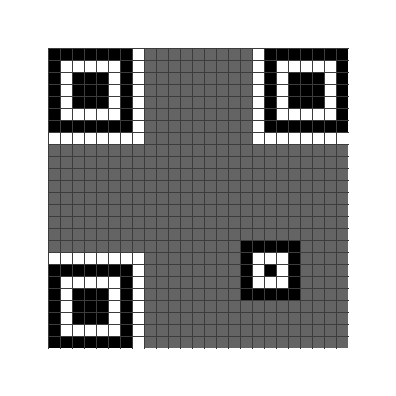
Các vị trí mà tại đó các mẫu căn chỉnh phải được đặt được xác định trong bảng vị trí mẫu căn chỉnh . Các số sẽ được sử dụng làm tọa độ CẢ hàng và cột. Ví dụ: Phiên bản 2 có các số 6 và 18. Điều này có nghĩa là MÔ -ĐUN TRUNG TÂM của các mẫu căn chỉnh phải được đặt tại (6, 6), (6, 18), (18, 6) và (18, 18) .

TUY NHIÊN , các mẫu căn chỉnh phải được đưa vào ma trận SAU KHI các mẫu công cụ tìm kiếm và dấu phân cách đã được đặt và các mẫu căn chỉnh KHÔNG PHẢI chồng lên các mẫu công cụ tìm kiếm hoặc dấu phân cách. Các hình ảnh sau đây hiển thị mã phiên bản 2, được mô tả trong đoạn trước là có các mẫu căn chỉnh tập trung vào (6, 6), (6, 18), (18, 6) và (18, 18). Tuy nhiên, như hình ảnh bên trái hiển thị, không được đặt các mẫu căn chỉnh được đánh dấu màu đỏ trong ma trận vì chúng chồng lên các mẫu tìm kiếm và dấu phân cách. Các mẫu căn chỉnh chồng lên các mẫu tìm kiếm hoặc dấu phân cách sẽ bị loại bỏ khỏi ma trận.

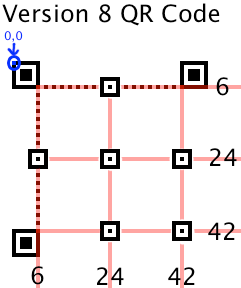
**KHÔNG ĐÚNG**

****

**CHÍNH XÁC**

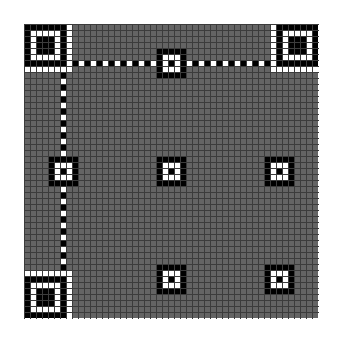
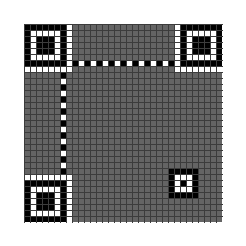
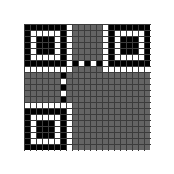
****

Để minh họa thêm về vị trí mẫu căn chỉnh, hình ảnh sau đây hiển thị các mẫu căn chỉnh cho mã QR phiên bản 8. Bảng vị trí mẫu căn chỉnh liệt kê 6, 24 và 42 làm vị trí mẫu căn chỉnh cho phiên bản 8. Tất cả sự kết hợp của ba số này được sử dụng làm tọa độ cho các mẫu căn chỉnh.

****

*Bước 4: Thêm các mẫu thời gian*

Các mẫu thời gian là hai đường, một đường ngang và một đường dọc, gồm các module sáng và tối xen kẽ. Mẫu thời gian ngang được đặt trên hàng thứ 6 của mã QR giữa các dấu phân cách. Mẫu thời gian dọc được đặt trên cột thứ 6 của mã QR giữa các dấu phân cách. Các mẫu thời gian luôn bắt đầu và kết thúc bằng module tối. Các mẫu căn chỉnh có thể chồng lên các mẫu thời gian vì các module sáng và tối của chúng luôn trùng với các module sáng và tối của các mẫu thời gian, như có thể thấy trong hình ảnh bên phải. Các hình ảnh sau đây hiển thị các mẫu thời gian trên các phiên bản mã QR khác nhau.



*Bước 5: Thêm module tối và khu vực dành riêng*

Đã đến lúc thêm các bit dữ liệu vào ma trận mã QR. Tuy nhiên, trước khi thực hiện điều đó, module tối phải được thêm vào và có các vùng của ma trận phải được dành riêng cho các bit định dạng và phiên bản, sẽ được thêm vào sau.

*Module tối*

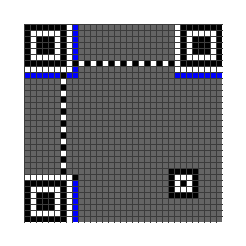
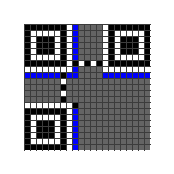
Như đã đề cập trước đó trên trang này, tất cả các mã QR đều có module tối bên cạnh mẫu công cụ tìm phía dưới bên trái. Cụ thể hơn, module tối luôn nằm ở tọa độ ([(4 × V) + 9], 8) trong đó V là phiên bản của mã QR.

*Dự trữ khu vực thông tin định dạng*

Một dải module bên cạnh các dấu phân cách phải được dành riêng cho vùng thông tin định dạng như sau:

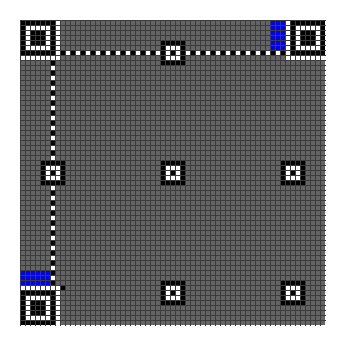
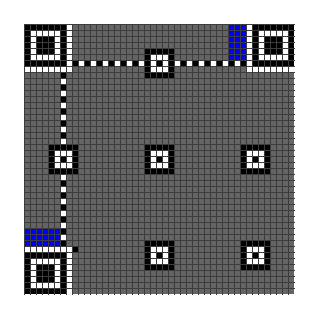
* Gần mẫu công cụ tìm trên cùng bên trái, dải một module phải được dành riêng bên dưới và bên phải của dấu phân cách.
* Gần mẫu công cụ tìm trên cùng bên phải, dải một module phải được dành riêng bên dưới dấu phân cách.
* Gần mẫu công cụ tìm phía dưới bên trái, dải một module phải được dành riêng ở bên phải dấu phân cách.

Những hình ảnh sau đây hiển thị các khu vực dành riêng có màu xanh lam. Chúng luôn được đặt dọc theo các dải phân cách, bất kể mã QR là phiên bản nào.



*Dự trữ khu vực thông tin phiên bản*

Mã QR phiên bản 7 trở lên phải chứa hai vùng đặt bit thông tin phiên bản. Các khu vực này là một khối 6x3 phía trên mẫu công cụ tìm phía dưới bên trái và một khối 3x6 ở bên trái của mẫu công cụ tìm phía trên bên phải. Các hình ảnh sau đây hiển thị vị trí của các khu vực dành riêng có màu xanh lam.



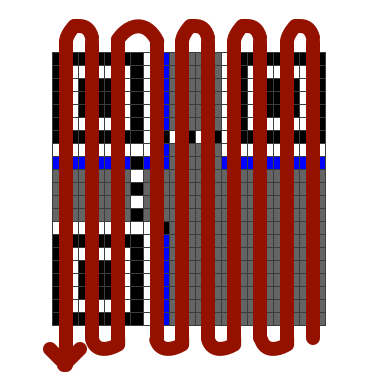
*Bước 6: Đặt các bit dữ liệu*

Bây giờ là lúc thêm các bit dữ liệu vào ma trận QR. Các bit được đặt trong một mẫu cụ thể.

*Mô hình vị trí*

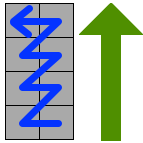
Các bit dữ liệu được đặt bắt đầu từ dưới cùng bên phải của ma trận và tiến lên phía trên trong một cột rộng 2 module. Sử dụng pixel trắng cho 0 và pixel đen cho 1. Khi cột lên đến đỉnh, cột 2 module tiếp theo sẽ bắt đầu ngay bên trái của cột trước đó và tiếp tục đi xuống. Bất cứ khi nào cột hiện tại chạm tới cạnh của ma trận, hãy chuyển sang cột 2 module tiếp theo và đổi hướng. Nếu gặp phải mẫu chức năng hoặc vùng dành riêng, bit dữ liệu sẽ được đặt vào module không sử dụng tiếp theo.

Hình ảnh sau đây hiển thị mẫu đặt các bit dữ liệu trong mã QR. Lưu ý rằng khi đạt đến mẫu thời gian dọc, cột tiếp theo sẽ bắt đầu ở bên trái của mẫu đó.

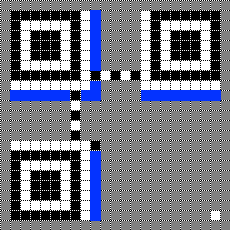


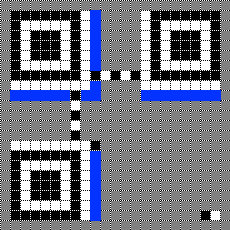
*Vị trí trở lên*

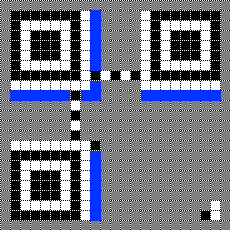
Hình ảnh sau đây cho thấy thứ tự đặt các bit dữ liệu khi cột đi lên.

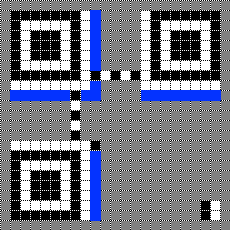


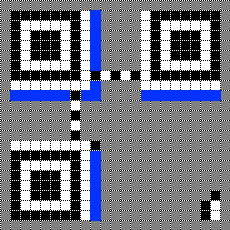
Ví dụ sau minh họa vị trí của các bit trong cột hướng lên đầu tiên.

  
Đã thêm pixel đầu tiên

  
Đã thêm pixel thứ hai

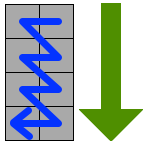
  
Đã thêm pixel thứ ba

  
Đã thêm pixel thứ tư

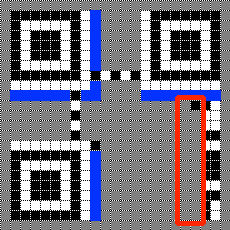
  
Đã thêm pixel thứ năm

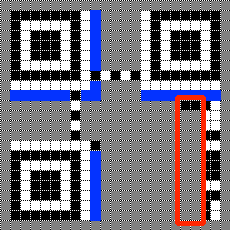
*Vị trí hướng xuống*

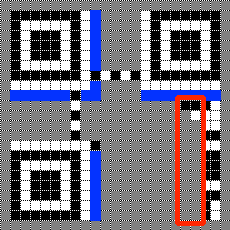
Hình ảnh sau đây cho thấy thứ tự đặt các bit dữ liệu khi cột đi xuống.

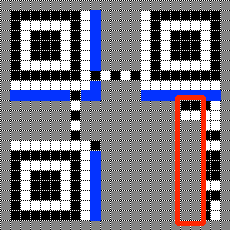


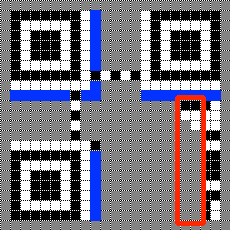
Ví dụ sau đây cho thấy các bit dữ liệu được đặt trong một cột hướng xuống.

  
Đã thêm pixel hướng xuống số 1

  
Đã thêm pixel hướng xuống số 2

  
Đã thêm pixel hướng xuống số 3

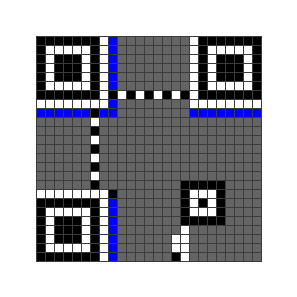
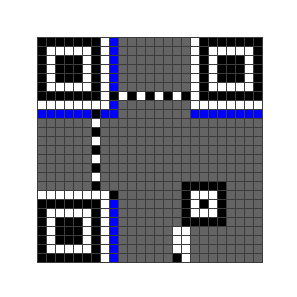
  
Đã thêm pixel #4 hướng xuống

  
Đã thêm pixel hướng xuống #5

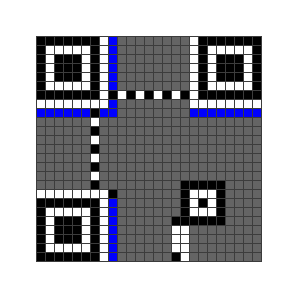
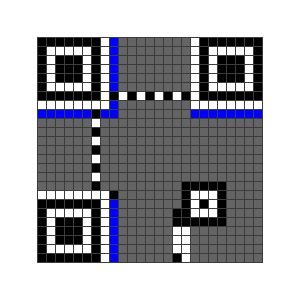
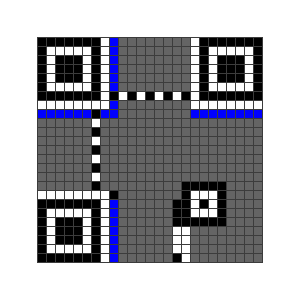
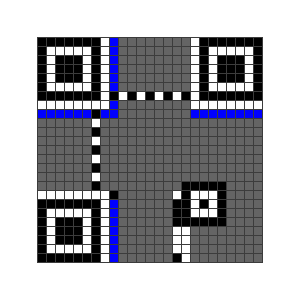
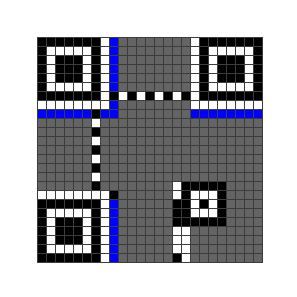
*Bỏ qua các mẫu hàm*

Khi gặp một mẫu chức năng, hãy bỏ qua mọi module bị chiếm dụng cho đến khi bạn đến module không sử dụng tiếp theo.

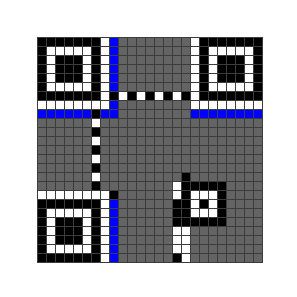
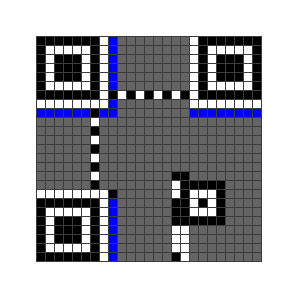
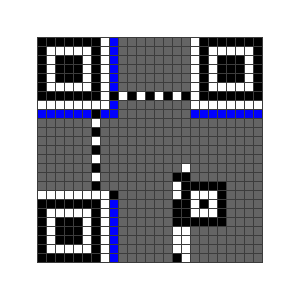
Trong các hình ảnh sau đây, các bit dữ liệu bên dưới mẫu căn chỉnh đang tiến hành theo cột hướng lên trên. Lưu ý rằng cột chồng lên mẫu căn chỉnh.

Khi đạt đến mẫu căn chỉnh, chỉ cần bỏ qua các module là một phần của mẫu căn chỉnh và tiếp tục đi lên.

Sau đó tiếp tục thêm module vào cột theo cách thông thường.

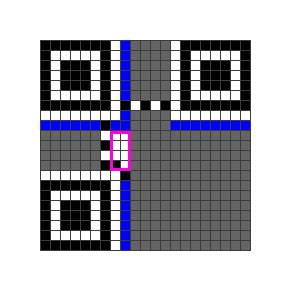
  
  


Tóm lại, luôn tiến hành bình thường dọc theo các cột, bỏ qua bất kỳ module nào được sử dụng bởi các mẫu chức năng hoặc khu vực dành riêng. Ngoại lệ duy nhất là mẫu thời gian dọc, như được giải thích bên dưới.

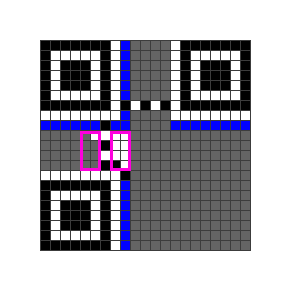
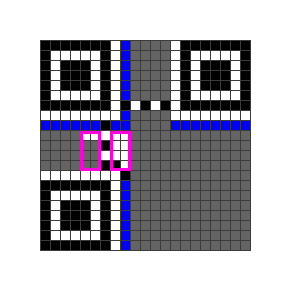
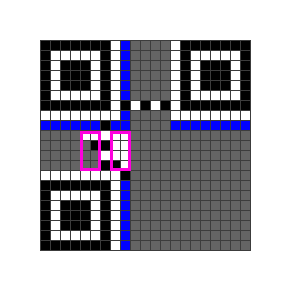
*Ngoại lệ: Mẫu thời gian dọc*

Mẫu thời gian dọc là ngoại lệ duy nhất cho quy tắc này. Khi đạt đến mẫu thời gian dọc, luôn bắt đầu cột tiếp theo ở bên trái của cột đó. Không có cột nào được chồng lên mẫu thời gian dọc.

Trong hình ảnh sau, cột trước đó (hướng lên trên) được đánh dấu. Phần còn lại của các module phía trên cột này được chiếm bởi các mẫu chức năng và các khu vực dành riêng, vì vậy cột tiếp theo phải được bắt đầu, đi xuống.



Như được hiển thị trong các hình ảnh sau, cột tiếp theo bắt đầu ở bên trái của mẫu thời gian dọc. Cột không chồng lên mẫu thời gian dọc.

**Che giấu dữ liệu**

Bây giờ các module đã được đặt trong ma trận , mẫu mặt nạ tốt nhất phải được xác định. Mẫu mặt nạ thay đổi module nào tối và module nào sáng theo một quy tắc cụ thể. Mục đích của bước này là sửa đổi mã QR để giúp trình đọc mã QR dễ dàng quét nhất có thể.

*Thuật ngữ: Mặt nạ*

Nếu một module trong mã QR bị "che đậy", điều này chỉ có nghĩa là nếu là module sáng thì nên đổi thành module tối, còn nếu là module tối thì nên đổi thành module sáng. Nói cách khác, mặt nạ đơn giản có nghĩa là chuyển đổi màu sắc của module.

*Tổng quan về các mẫu mặt nạ*

Đặc tả mã QR xác định tám [mẫu mặt nạ](https://www.thonky.com/qr-code-tutorial/mask-patterns) có thể được áp dụng cho mã QR. Ví dụ: đối với mẫu mặt nạ số 1, mọi hàng đánh số chẵn trong ma trận QR đều bị che và đối với mẫu mặt nạ số 2, mọi cột thứ ba trong ma trận QR đều bị che.

Vui lòng tham khảo trang [mẫu mặt nạ](https://www.thonky.com/qr-code-tutorial/mask-patterns) để biết thêm chi tiết về tám mẫu mặt nạ.

*Mặt nạ gì*

Các mẫu mặt nạ CHỈ phải được áp dụng cho **các module dữ liệu và module sửa lỗi** . Nói cách khác:

* Không che dấu các mẫu chức năng (mẫu tìm kiếm, mẫu định giờ, dấu phân cách, mẫu căn chỉnh)
* Không che dấu các khu vực dành riêng (khu vực thông tin định dạng, khu vực thông tin phiên bản)

*Xác định mặt nạ tốt nhất*

Sau khi áp dụng mẫu mặt nạ cho ma trận QR, mẫu mặt nạ đó sẽ bị phạt dựa trên bốn điều kiện đánh giá được xác định trong đặc tả mã QR. Bộ mã hóa mã QR phải áp dụng tất cả tám mẫu mặt nạ và đánh giá từng mẫu. Bất kỳ mẫu mặt nạ nào dẫn đến điểm phạt thấp nhất là mẫu mặt nạ phải được sử dụng cho đầu ra cuối cùng.

*Cách đánh giá các khu vực dành riêng*

Lưu ý rằng toàn bộ ma trận (bao gồm các mẫu chức năng và các vùng dành riêng) đều được đánh giá, mặc dù mặt nạ chỉ được áp dụng cho các module sửa lỗi và dữ liệu.

*Bốn quy tắc hình phạt*

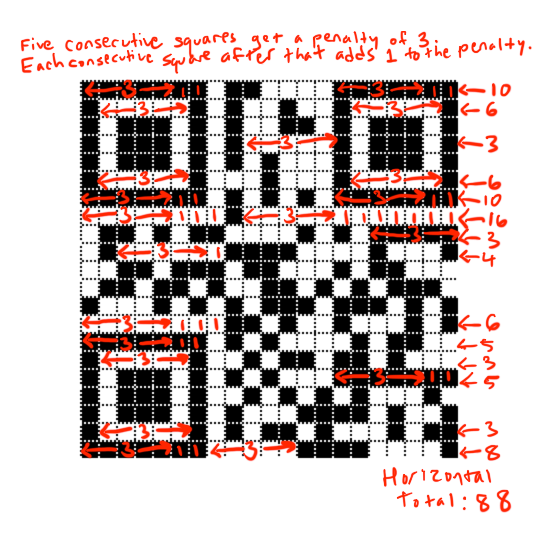
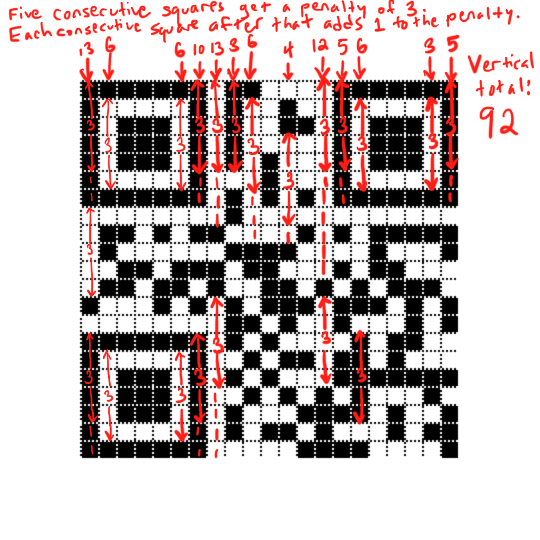
Bốn quy định về hình phạt có thể được tóm tắt như sau:

* Quy tắc đầu tiên đưa ra hình phạt cho mã QR đối với mỗi nhóm gồm năm module cùng màu trở lên trong một hàng (hoặc cột).
* Quy tắc thứ hai đưa ra hình phạt cho mã QR đối với mỗi vùng 2x2 của các module cùng màu trong ma trận.
* Quy tắc thứ ba quy định mức phạt lớn cho mã QR nếu có các mẫu trông giống với mẫu của công cụ tìm kiếm.
* Quy tắc thứ tư đưa ra hình phạt cho mã QR nếu hơn một nửa số module có màu tối hoặc sáng, với hình phạt lớn hơn cho sự khác biệt lớn hơn.

*Điều kiện đánh giá số 1*

Đối với điều kiện đánh giá đầu tiên, hãy kiểm tra từng hàng một. Nếu có năm module liên tiếp cùng màu, hãy cộng thêm 3 vào hình phạt. Nếu có nhiều module cùng màu hơn sau năm module đầu tiên, hãy thêm 1 module cho mỗi module bổ sung có cùng màu. Sau đó, kiểm tra từng cột một, kiểm tra tình trạng tương tự. Cộng tổng số theo chiều ngang và chiều dọc để có được điểm phạt số 1.

Các hình ảnh sau đây minh họa quá trình đánh giá mã QR theo cách này. Trong ví dụ này, mức phạt theo chiều ngang là 92 và mức phạt theo chiều dọc là 88. Do đó, điểm phạt số 1 là 92 + 88 = 180.

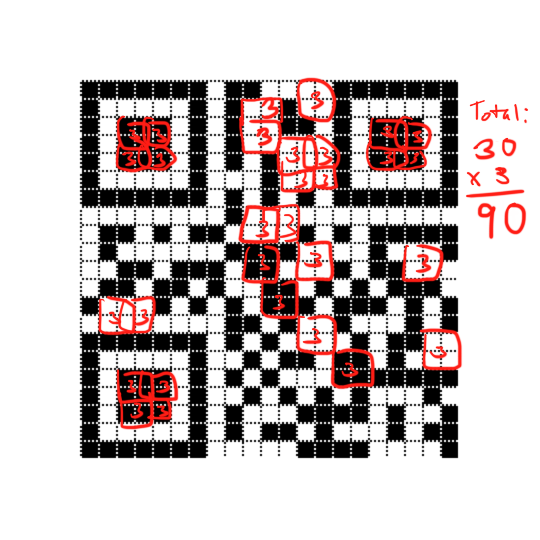
  


*Điều kiện đánh giá số 2*

Đối với điều kiện đánh giá thứ hai, hãy tìm các vùng có cùng màu có kích thước module ít nhất là 2x2 hoặc lớn hơn. Đặc tả mã QR cho biết rằng đối với khối màu đồng nhất có kích thước m × n, điểm phạt là 3 × (m - 1) × (n - 1). Tuy nhiên, đặc tả mã QR không chỉ rõ cách tính mức phạt khi có nhiều cách chia các khối đồng màu.

Do đó, thay vì tìm kiếm các khối đồng màu lớn hơn 2x2, bạn chỉ cần thêm 3 vào điểm phạt cho mỗi khối 2x2 cùng màu trong mã QR, đảm bảo tính các khối 2x2 chồng chéo. Ví dụ: một khối 3x2 cùng màu phải được tính là hai khối 2x2, khối này chồng lên khối kia.

Hình ảnh sau đây minh họa cách tính quy tắc phạt số 2.



*Điều kiện đánh giá số 3*

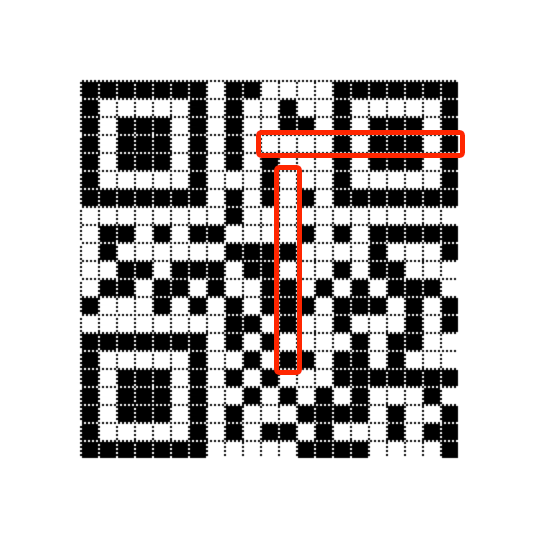
Quy tắc hình phạt thứ ba tìm kiếm các mẫu tối-sáng-tối-tối-tối-sáng-tối có bốn module ánh sáng ở hai bên. Nói cách khác, nó tìm kiếm bất kỳ mẫu nào trong hai mẫu sau:

https://www.thonky.com/qr-code-tutorial/penalty-3-pattern1.png

HOẶC

https://www.thonky.com/qr-code-tutorial/penalty-3-pattern2.png

Mỗi lần tìm thấy mẫu này, hãy cộng 40 vào điểm phạt. Trong ví dụ dưới đây, có hai mẫu như vậy. Do đó, điểm phạt số 3 là 80.

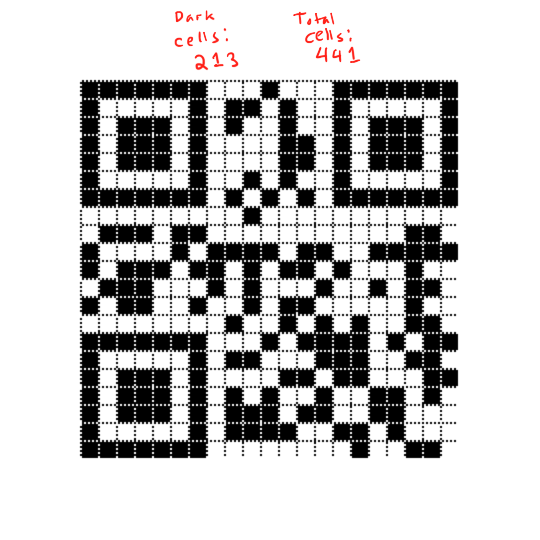


*Điều kiện đánh giá số 4*

Điều kiện đánh giá cuối cùng dựa trên tỷ lệ giữa module sáng và module tối. Để tính quy tắc phạt này, hãy thực hiện các bước sau:

1. Đếm tổng số module trong ma trận.
2. Đếm xem có bao nhiêu module tối trong ma trận.
3. Tính phần trăm module tối trong ma trận: (darkmodules / Totalmodules) × 100
4. Xác định bội số trước và bội số tiếp theo của 5 của phần trăm này. Ví dụ: đối với 43 phần trăm, **bội số trước của 5 là 40** và **bội số tiếp theo của 5 là 45** .
5. Trừ 50 từ mỗi bội số của 5 và lấy giá trị tuyệt đối của kết quả. Ví dụ: |40 - 50| = |-10| = 10 và |45 - 50| = |-5| = 5.
6. Chia mỗi thứ này cho năm. Ví dụ: 10/5 = 2 và 5/5 = 1.
7. Cuối cùng, lấy số nhỏ nhất trong hai số và nhân với 10. Trong ví dụ này, số nhỏ hơn là 1 nên kết quả là 10. Đây là điểm phạt số 4.

Ví dụ khác, trong hình ảnh bên dưới, tổng số module là 441 và tổng số module tối là 213.



Phần trăm của module tối là (213/441) × 100 ≈ 48,2993

Bội số trước của 5 là 45 và bội số tiếp theo của 5 là 50.

Trừ 50 và lấy giá trị tuyệt đối của mỗi số:  
|45 - 50| = |-5| = 5  
|50 - 50| = 0

Chia mỗi số cho 5:  
5/5 = 1  
0/5 = 0

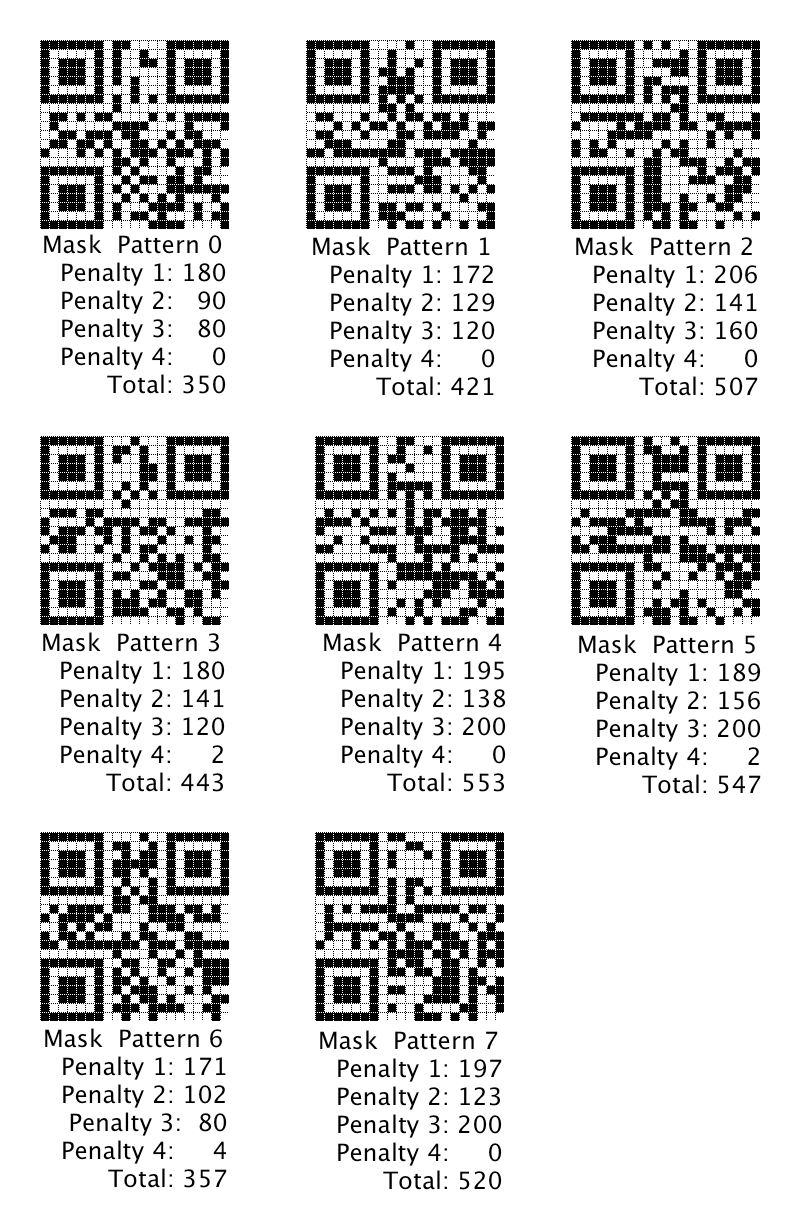
Số nhỏ nhất trong hai số đó là 0. Nhân với 10 thì vẫn bằng 0. Do đó, điểm phạt số 4 trong ví dụ này là 0.

*Thêm bốn điểm phạt đền*

Để hoàn thành việc đánh giá mã QR, hãy cộng bốn điểm phạt. Tổng số là điểm phạt tổng thể của mã QR.

*Chọn Điểm Phạt Thấp Nhất cho Tám Mẫu Mặt nạ*

Các hình ảnh sau đây hiển thị tám mã QR, một mã cho mỗi mẫu mặt nạ. Tất cả tám mã QR trong ví dụ này đều mã hóa cùng một dữ liệu. Như được hiển thị, mẫu mặt nạ có điểm phạt thấp nhất là mẫu mặt nạ 0. Do đó, trong ví dụ này, bộ mã hóa QR nên sử dụng mẫu mặt nạ 0 khi xuất mã QR cuối cùng.



**Thông tin về định dạng và phiên bản**

Bước cuối cùng để tạo Mã QR là tạo chuỗi định dạng và phiên bản, sau đó đặt chúng vào đúng vị trí trong mã QR. Phần này giải thích cách tạo chuỗi định dạng và phiên bản, đồng thời minh họa vị trí chúng được đặt trong mã QR.

*Chuỗi định dạng*

Chuỗi thông tin định dạng mã hóa mức sửa lỗi và mẫu mặt nạ nào đang được sử dụng trong mã QR hiện tại. Vì có bốn mức sửa lỗi có thể có (L, M, Q và H) và tám mẫu mặt nạ có thể có nên có 32 (4 nhân 8) chuỗi thông tin định dạng có thể có. Phần tiếp theo giải thích cách tạo các chuỗi định dạng này. Để biết danh sách đầy đủ 32 chuỗi định dạng, vui lòng tham khảo bảng chuỗi định dạng.

*Tạo chuỗi định dạng*

Chuỗi định dạng luôn dài 15 bit. Để tạo chuỗi, trước tiên bạn tạo chuỗi 5 bit mã hóa mức sửa lỗi và mẫu mặt nạ được sử dụng trong mã QR này. Sau đó, bạn sử dụng năm bit đó để tạo ra mười bit sửa lỗi. Mười lăm bit kết quả được XOR với mẫu mặt nạ 101010000010010. Quá trình này được giải thích chi tiết bên dưới.

*Các bit sửa lỗi*

Bước đầu tiên để tạo chuỗi định dạng là lấy hai bit xác định mức sửa lỗi được sử dụng trong mã QR. Bảng sau đây hiển thị các chuỗi bit cho từng mức sửa lỗi.

|  |  |  |
| --- | --- | --- |
| **Mức độ sửa lỗi** | **Chút ít** | **Số nguyên tương đương** |
| L | 01 | 1 |
| M | 00 | 0 |
| Q | 11 | 3 |
| H | 10 | 2 |

Lưu ý rằng các số không theo thứ tự 0, 1, 2, 3 trong bảng.

Các bit mẫu mặt nạ

Đối với các mẫu mặt nạ, hãy tham khảo trang [mẫu mặt nạ mã QR](https://www.thonky.com/qr-code-tutorial/mask-patterns) để tìm số mặt nạ đi kèm với từng mẫu. Chuyển đổi số thành chuỗi nhị phân ba bit.

Ví dụ về chuỗi định dạng 5 bit

Ví dụ: nếu chúng ta đã sử dụng mức sửa lỗi L và mẫu mặt nạ 4, chuỗi nhị phân 5 bit sẽ được tạo như sau:

01 (chỉ báo mức sửa lỗi L)  
  
100 (nhị phân cho 4, tức là mẫu mặt nạ 4)  
  
Kết quả: 01100

*Tạo bit sửa lỗi cho chuỗi định dạng*

Bây giờ chúng ta có 5 bit cho chuỗi định dạng, chúng ta phải sử dụng nó để tạo ra 10 bit sửa lỗi. Bước này sử dụng Công cụ sửa lỗi Reed-Solomon, nhưng dễ dàng hơn một chút vì trong trường hợp này, đa thức chứa không quá mười lăm số hạng và hệ số của chúng đều là 1 hoặc 0.

**Lấy đa thức sinh**

Khi tạo từ mã sửa lỗi của chuỗi định dạng, đặc tả mã QR yêu cầu sử dụng đa thức tạo sau:  
x 10 + x 8 + x 5 + x 4 + x 2 + x + 1

Chúng ta có thể chuyển đổi chuỗi này thành chuỗi nhị phân bằng cách chỉ lấy các hệ số của mỗi số hạng. Hệ số của x 10 là 1, hệ số của x 9 là 0 vì x 9 không có trong đa thức, v.v.

Nói cách khác, chuỗi nhị phân đại diện cho đa thức tạo cho bước này là  
10100110111

**Tính số bit sửa lỗi**

Bước tiếp theo là chia các bit chuỗi định dạng (01100 từ bước trước) cho đa thức tạo (10100110111 từ bước trước).

Để thực hiện việc này, trước tiên hãy tạo chuỗi 15 bit bằng cách đặt mười số 0 ở bên phải chuỗi định dạng, như sau:  
01100 -> 0110000000000000

Bây giờ hãy xóa mọi số 0 ở phía bên trái:  
011000000000000 -> 11000000000000

Bây giờ chúng ta thực hiện phép chia. Các bước (được mô tả chi tiết hơn bên dưới) là:  
1. Thêm số 0 vào chuỗi đa thức của trình tạo ở bên phải để làm cho nó có cùng độ dài với chuỗi định dạng hiện tại.  
2. XOR chuỗi đa thức tạo đệm với chuỗi định dạng hiện tại.  
3. Xóa số 0 ở phía bên trái của kết quả.

Chúng ta phải chia các đa thức cho đến khi chuỗi định dạng kết quả dài 10 bit hoặc ít hơn. Do đó, trước mỗi bước chia, hãy kiểm tra để đảm bảo rằng chuỗi định dạng hiện tại là 11 bit hoặc dài hơn (11 bit là độ dài của đa thức tạo). Chuỗi hiện tại của chúng tôi, 11000000000000, dài 14 bit.

**Phân chia đầu tiên**

Đầu tiên, đệm chuỗi đa thức của trình tạo ở bên phải bằng các số 0 để làm cho nó có cùng độ dài với chuỗi định dạng hiện tại:  
10100110111 11000000000000 10100110111000

Bây giờ, XOR chuỗi định dạng với đa thức tạo đệm:  
11000000000000 ^ 10100110111000 = 01100110111000

Và xóa số 0 ở phía bên trái của kết quả:  
01100110111000 -> 1100110111000

**Bộ phận thứ hai**

Chuỗi kết quả 1100110111000 dài 13 bit, do đó, chuỗi đó dài 11 bit hoặc dài hơn nên chúng ta có thể tiếp tục.

Đệm đa thức sinh ở bên phải để tạo cùng độ dài với chuỗi định dạng hiện tại:  
10100110111 1100110111000 1010011011100

XOR chuỗi định dạng hiện tại với đa thức tạo đệm: 1100110111000 ^ 1010011011100 = 110101100100

**Giải hạng Ba**

Chuỗi kết quả 110101100100 dài 12 bit, do đó, chuỗi đó dài 11 bit hoặc dài hơn nên chúng ta có thể tiếp tục.

Đệm đa thức sinh ở bên phải để tạo cùng độ dài với chuỗi định dạng hiện tại:  
10100110111 110101100100 101001101110

XOR chuỗi định dạng hiện tại với đa thức tạo đệm: 110101100100 ^ 101001101110 = 011100001010

Và xóa số 0 ở phía bên trái của kết quả:  
011100001010 -> 11100001010

**Giải hạng tư**

Chuỗi kết quả 11100001010 dài 11 bit, vì vậy chuỗi đó dài 11 bit hoặc dài hơn nên chúng ta có thể tiếp tục.

Không cần phải đệm đa thức của trình tạo vì chuỗi định dạng bây giờ có cùng độ dài.

XOR chuỗi định dạng hiện tại với đa thức tạo: 11100001010 ^ 10100110111 = 1000111101

Kết quả dài 10 bit nên chúng ta đã hoàn thành bước chia. **Nếu kết quả nhỏ hơn 10 bit, chúng ta sẽ đệm nó ở TRÁI bằng các số 0 để làm cho nó dài 10 bit.**

**Đặt các bit định dạng và sửa lỗi lại với nhau**

Tạo một chuỗi bằng cách sử dụng năm bit ban đầu của thông tin định dạng (chỉ báo mức sửa lỗi và chỉ báo mẫu mặt nạ), theo sau là các bit sửa lỗi mà chúng ta vừa tạo.

Chuỗi định dạng năm bit: 01100  
Chuỗi sửa lỗi mười bit từ bước chia: 1000111101  
Chuỗi kết hợp: 011001000111101

**XOR với chuỗi mặt nạ**

Đặc tả mã QR cho XOR kết quả bằng chuỗi nhị phân sau: 101010000010010

011001000111101 ^ 101010000010010 = 110011000101111

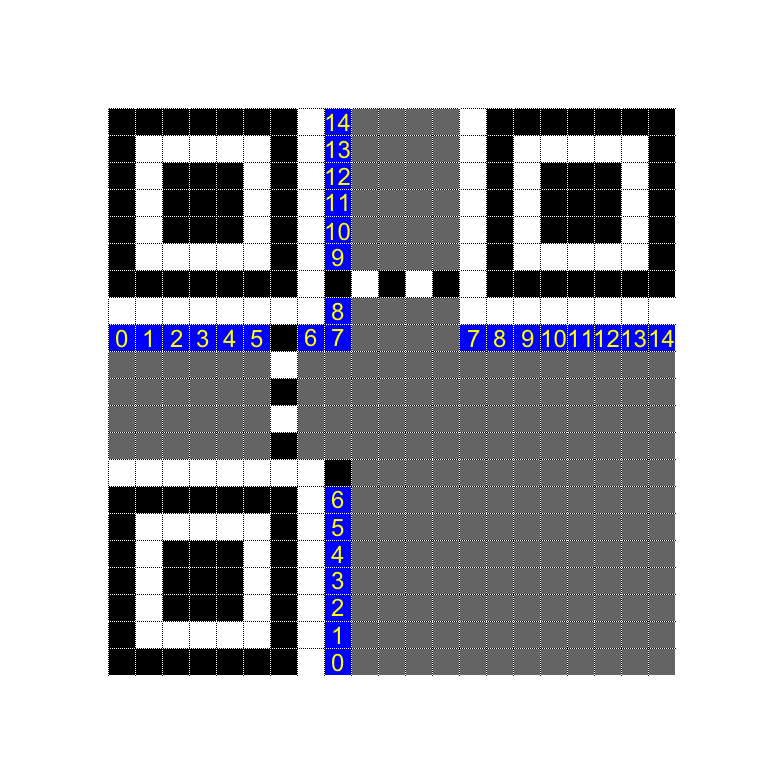
Chuỗi **định dạng cuối cùng** cho mã có mức sửa lỗi L và mẫu mặt nạ 4 là 110011000101111

Để biết danh sách tất cả 32 chuỗi định dạng, hãy tham khảo [bảng chuỗi định dạng](https://www.thonky.com/qr-code-tutorial/format-version-tables) .

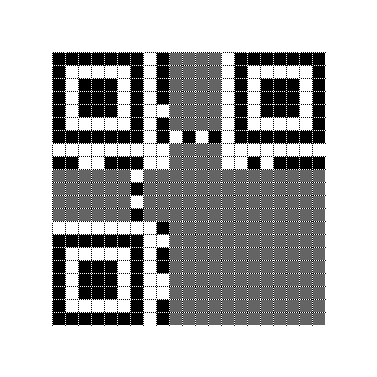
*Đưa Chuỗi Định Dạng vào Mã QR*

Chuỗi thông tin định dạng được đặt bên dưới các mẫu công cụ tìm kiếm trên cùng và ở bên phải của các mẫu công cụ tìm kiếm ngoài cùng bên trái, như minh họa trong hình ảnh bên dưới. **Số 0 trong hình ảnh đề cập đến bit quan trọng nhất của chuỗi định dạng và số 14 đề cập đến bit ít quan trọng nhất.**Nói cách khác, sử dụng chuỗi định dạng 110011000101111 từ ví dụ trên, các số trong hình ảnh bên dưới tương ứng với các bit chuỗi định dạng như sau:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |



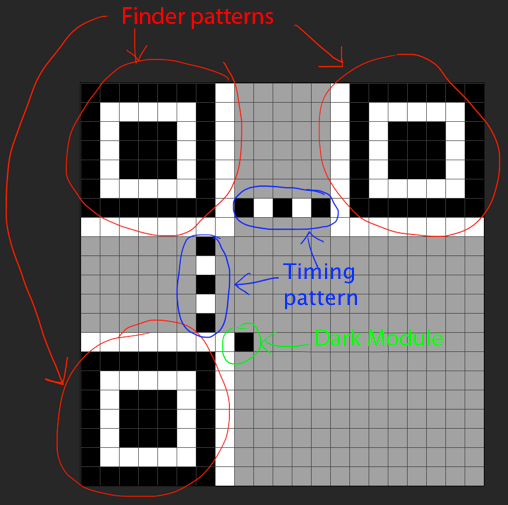
Hình ảnh bên dưới hiển thị chuỗi định dạng mẫu, 110011000101111, trong mã QR phiên bản 1.



*Mô-đun tối*

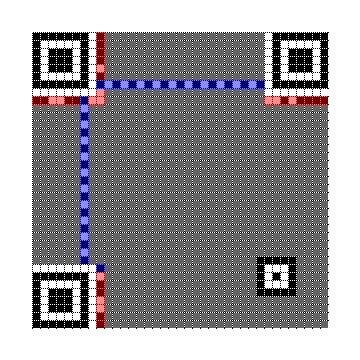
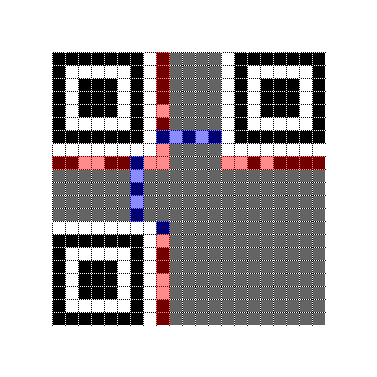
Mỗi mã QR phải có một pixel tối hay còn gọi là mô-đun tối tại tọa độ (8, 4×version + 9). Tức là tọa độ y của mô-đun tối là  
phiên bản 1: 4×1 + 9 = 13  
phiên bản 2: 4×2 + 9 = 17  
phiên bản 3: 4×3 + 9 = 21  
  
, v.v.

Điều này có nghĩa là mô-đun tối luôn ở bên phải góc trên cùng bên phải của mẫu công cụ tìm phía dưới bên trái. Điều này được thể hiện trong hình ảnh dưới đây.



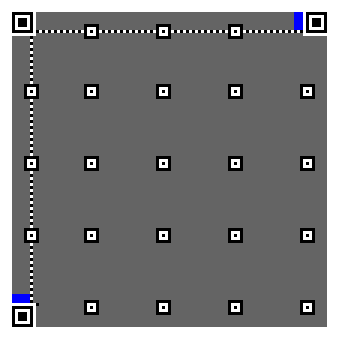
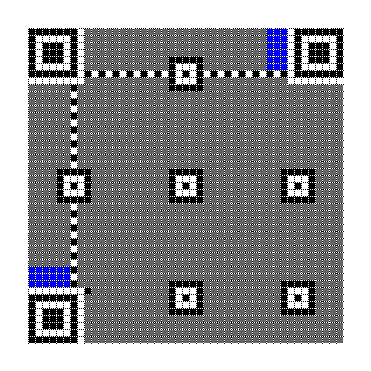
*Đối với mã lớn hơn*

Bất kể kích thước của mã QR là bao nhiêu, các bit chuỗi định dạng được đặt bên dưới các mẫu công cụ tìm kiếm trên cùng và ở bên phải của các mẫu công cụ tìm kiếm ngoài cùng bên trái, như minh họa bên dưới. Lưu ý rằng cả hai hình ảnh đều có cùng chuỗi định dạng, 110011000101111, từ ví dụ trước đó trên trang này. Chuỗi định dạng được đánh dấu bằng màu đỏ và mẫu thời gian và mô-đun tối được đánh dấu bằng màu xanh lam.



*Thông tin phiên bản*

Nếu Mã QR là phiên bản 7 trở lên, bạn phải bao gồm chuỗi thông tin phiên bản 18 bit ở góc dưới cùng bên trái và trên cùng bên phải của mã QR. (Để biết danh sách đầy đủ tất cả các chuỗi thông tin phiên bản có thể có, hãy tham khảo [bảng định dạng và phiên bản](https://www.thonky.com/qr-code-tutorial/format-version-tables) .) Các vùng thông tin phiên bản là các hình chữ nhật màu xanh 6x3 được hiển thị trong các hình ảnh bên dưới. Thông tin phiên bản được đặt bên cạnh các mẫu công cụ tìm kiếm cho dù mã QR có lớn đến đâu. Hình ảnh bên trái là mã QR phiên bản 7 và hình ảnh bên phải là mã QR phiên bản 22.



*Tạo chuỗi thông tin phiên bản*

Đặc tả Mã QR cho biết sử dụng mã Golay (18, 6) cho chuỗi thông tin phiên bản. Như vậy, chuỗi thông tin phiên bản là chuỗi 18 bit bao gồm chuỗi nhị phân 6 bit mã hóa phiên bản QR, theo sau là chuỗi 12 bit sửa lỗi. Toàn bộ chuỗi dài 18 bit.

*Lấy đa thức sinh*

Đặc tả mã QR yêu cầu sử dụng đa thức tạo sau cho bước này:  
x 12 + x 11 + x 10 + x 9 + x 8 + x 5 + x 2 + 1

Như đã giải thích trong phần chuỗi định dạng trước đó trên trang này, chúng ta có thể biểu diễn đa thức của trình tạo này bằng chuỗi nhị phân sau:  
1111100100101

*Thực hiện phép chia*

Từ đây, chúng ta có thể thực hiện theo các bước phân chia tương tự mà chúng ta đã sử dụng để tạo chuỗi thông tin định dạng, ngoại trừ trong trường hợp này, chúng ta đệm các chuỗi ban đầu thành 18 bit thay vì 15 và chúng ta dừng khi chuỗi bit hiện tại là 12 hoặc ít hơn bit dài, thay vì 10 hoặc ít hơn.

Bắt đầu bằng cách tạo một chuỗi nhị phân sáu bit đại diện cho số phiên bản. Ví dụ: đối với mã phiên bản 7, mã nhị phân 6 bit tương đương với 7 là:  
000111

Biến chuỗi này thành chuỗi 18 bit bằng cách đệm bên phải bằng 0:  
000111000000000000

Và xóa số 0 ở phía bên trái:  
111000000000000

Bây giờ hãy đệm đa thức sinh ở bên phải bằng các số 0 để có cùng độ dài.  
111000000000000 1111100100101 111110010010100

XOR chuỗi phiên bản và đa thức tạo đệm:  
1110000000000 ^ 111110010010100 = 110010010100

Chuỗi này đã có độ dài yêu cầu là 12 nên không cần chia thêm nữa. Giống như chuỗi thông tin định dạng, **nếu kết quả nhỏ hơn 12 thì nó phải được đệm ở TRÁI bằng các số 0 để tạo thành chuỗi dài 12 bit** .

Cuối cùng, đặt chuỗi phiên bản sáu bit gốc ở bên trái kết quả ở bước cuối cùng.  
chuỗi phiên bản: 000111  
chuỗi sửa lỗi từ phía trên: 110010010100  
chuỗi thông tin phiên bản cuối cùng: 000111110010010100

Để biết danh sách đầy đủ tất cả các chuỗi thông tin phiên bản có thể có, hãy tham khảo [bảng định dạng và phiên bản](https://www.thonky.com/qr-code-tutorial/format-version-tables) .

*Đặt chuỗi phiên bản vào mã QR*

Có hai khu vực hình chữ nhật phải đặt chuỗi thông tin phiên bản: một ở phía dưới bên trái và một ở phía trên bên phải.

Khối thông tin phiên bản dưới cùng bên trái

Khối thông tin phiên bản phía dưới bên trái cao 3 pixel và rộng 6 pixel. Bảng sau giải thích cách sắp xếp các bit của chuỗi thông tin phiên bản ở vùng thông tin phiên bản phía dưới bên trái. Số 0 đại diện cho bit RIGHTmost (ít quan trọng nhất) của chuỗi thông tin phiên bản và số 17 đại diện cho bit LEFTmost (quan trọng nhất) của chuỗi thông tin phiên bản.

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

*Khối thông tin phiên bản trên cùng bên phải*

Khối thông tin phiên bản trên cùng bên phải rộng 3 pixel và cao 6 pixel. Bảng sau giải thích cách sắp xếp các bit của chuỗi thông tin phiên bản ở vùng thông tin phiên bản trên cùng bên phải. Số 0 đại diện cho bit RIGHTmost (ít quan trọng nhất) của chuỗi thông tin phiên bản và số 17 đại diện cho bit LEFTmost (quan trọng nhất) của chuỗi thông tin phiên bản.

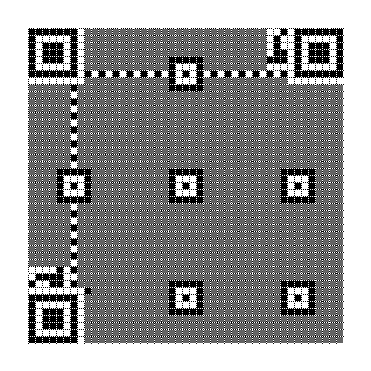
|  |  |  |
| --- | --- | --- |
| 00 | 01 | 02 |
| 03 | 04 | 05 |
| 06 | 07 | 08 |
| 09 | 10 | 11 |
| 12 | 13 | 14 |
| 15 | 16 | 17 |

Lấy chuỗi thông tin phiên bản 7, 000111110010010100 làm ví dụ, các số trong bảng trên tương ứng với các bit như sau:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |

*Ví dụ về chuỗi thông tin phiên bản 7*

Hình ảnh sau đây là mã QR phiên bản 7, sử dụng chuỗi thông tin phiên bản 000111110010010100. Lưu ý rằng các vùng thông tin phiên bản được điền theo mẫu được mô tả trong các bảng trên.



*Xuất ma trận cuối cùng*

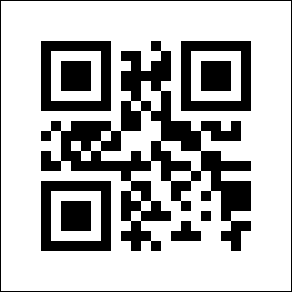
Bước cuối cùng, sau khi thêm thông tin định dạng và phiên bản vào ma trận QR, là xuất ra ma trận cuối cùng, sử dụng mẫu mặt nạ đã được xác định là có điểm phạt thấp nhất.

*Thêm vùng yên tĩnh*

Xin lưu ý rằng đặc tả mã QR yêu cầu ma trận QR được bao quanh bởi một vùng yên tĩnh: vùng mô-đun ánh sáng rộng 4 mô-đun.

*Đầu ra cuối cùng*

Hình ảnh sau đây là mã 1-Q của HELLO WORLD, được mã hóa ở chế độ chữ và số.



*Phần kết luận*

Điều này kết thúc hướng dẫn về mã QR. Tóm lại, mã QR được tạo theo các bước sau:

1. Xác định chế độ mã hóa nào sẽ sử dụng
2. Mã hóa dữ liệu
3. Tạo từ mã sửa lỗi
4. Các khối xen kẽ nếu cần thiết
5. Đặt các bit sửa lỗi và dữ liệu vào ma trận
6. Áp dụng các mẫu mặt nạ và xác định mẫu nào có mức phạt thấp nhất
7. Thêm thông tin định dạng và phiên bản

**Dung lượng ký tự theo phiên bản, chế độ và sửa lỗi**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **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 | 4 |
| 2 | 25x25 | L | 272 | 77 | 47 | 32 | 20 |
| M | 224 | 63 | 38 | 26 | 16 |
| Q | 176 | 48 | 29 | 20 | 12 |
| H | 128 | 34 | 20 | 14 | 8 |
| 3 | 29x29 | L | 440 | 127 | 77 | 53 | 32 |
| M | 352 | 101 | 61 | 42 | 26 |
| Q | 272 | 77 | 47 | 32 | 20 |
| H | 208 | 58 | 35 | 24 | 15 |
| 4 | 33x33 | L | 640 | 187 | 114 | 78 | 48 |
| M | 512 | 149 | 90 | 62 | 38 |
| Q | 384 | 111 | 67 | 46 | 28 |
| H | 288 | 82 | 50 | 34 | 21 |
| 5 | 37x37 | L | 864 | 255 | 154 | 106 | 65 |
| M | 688 | 202 | 122 | 84 | 52 |
| Q | 496 | 144 | 87 | 60 | 37 |
| H | 368 | 106 | 64 | 44 | 27 |
| 6 | 41x41 | L | 1,088 | 322 | 195 | 134 | 82 |
| M | 864 | 255 | 154 | 106 | 65 |
| Q | 608 | 178 | 108 | 74 | 45 |
| H | 480 | 139 | 84 | 58 | 36 |
| 7 | 45x45 | L | 1,248 | 370 | 224 | 154 | 95 |
| M | 992 | 293 | 178 | 122 | 75 |
| Q | 704 | 207 | 125 | 86 | 53 |
| H | 528 | 154 | 93 | 64 | 39 |
| 8 | 49x49 | L | 1,552 | 461 | 279 | 192 | 118 |
| M | 1,232 | 365 | 221 | 152 | 93 |
| Q | 880 | 259 | 157 | 108 | 66 |
| H | 688 | 202 | 122 | 84 | 52 |
| 9 | 53x53 | L | 1,856 | 552 | 335 | 230 | 141 |
| M | 1,456 | 432 | 262 | 180 | 111 |
| Q | 1,056 | 312 | 189 | 130 | 80 |
| H | 800 | 235 | 143 | 98 | 60 |
| 10 | 57x57 | L | 2,192 | 652 | 395 | 271 | 167 |
| M | 1,728 | 513 | 311 | 213 | 131 |
| Q | 1,232 | 364 | 221 | 151 | 93 |
| H | 976 | 288 | 174 | 119 | 74 |
| 11 | 61x61 | L | 2,592 | 772 | 468 | 321 | 198 |
| M | 2,032 | 604 | 366 | 251 | 155 |
| Q | 1,440 | 427 | 259 | 177 | 109 |
| H | 1,120 | 331 | 200 | 137 | 85 |
| 12 | 65x65 | L | 2,960 | 883 | 535 | 367 | 226 |
| M | 2,320 | 691 | 419 | 287 | 177 |
| Q | 1,648 | 489 | 296 | 203 | 125 |
| H | 1,264 | 374 | 227 | 155 | 96 |
| 13 | 69x69 | L | 3,424 | 1,022 | 619 | 425 | 262 |
| M | 2,672 | 796 | 483 | 331 | 204 |
| Q | 1,952 | 580 | 352 | 241 | 149 |
| H | 1,440 | 427 | 259 | 177 | 109 |
| 14 | 73x73 | L | 3,688 | 1,101 | 667 | 458 | 282 |
| M | 2,920 | 871 | 528 | 362 | 223 |
| Q | 2,088 | 621 | 376 | 258 | 159 |
| H | 1,576 | 468 | 283 | 194 | 120 |
| 15 | 77x77 | L | 4,184 | 1,250 | 758 | 520 | 320 |
| M | 3,320 | 991 | 600 | 412 | 254 |
| Q | 2,360 | 703 | 426 | 292 | 180 |
| H | 1,784 | 530 | 321 | 220 | 136 |
| 16 | 81x81 | L | 4,712 | 1,408 | 854 | 586 | 361 |
| M | 3,624 | 1,082 | 656 | 450 | 277 |
| Q | 2,600 | 775 | 470 | 322 | 198 |
| H | 2,024 | 602 | 365 | 250 | 154 |
| 17 | 85x85 | L | 5,176 | 1,548 | 938 | 644 | 397 |
| M | 4,056 | 1,212 | 734 | 504 | 310 |
| Q | 2,936 | 876 | 531 | 364 | 224 |
| H | 2,264 | 674 | 408 | 280 | 173 |
| 18 | 89x89 | L | 5,768 | 1,725 | 1,046 | 718 | 442 |
| M | 4,504 | 1,346 | 816 | 560 | 345 |
| Q | 3,176 | 948 | 574 | 394 | 243 |
| H | 2,504 | 746 | 452 | 310 | 191 |
| 19 | 93x93 | L | 6,360 | 1,903 | 1,153 | 792 | 488 |
| M | 5,016 | 1,500 | 909 | 624 | 384 |
| Q | 3,560 | 1,063 | 644 | 442 | 272 |
| H | 2,728 | 813 | 493 | 338 | 208 |
| 20 | 97x97 | L | 6,888 | 2,061 | 1,249 | 858 | 528 |
| M | 5,352 | 1,600 | 970 | 666 | 410 |
| Q | 3,880 | 1,159 | 702 | 482 | 297 |
| H | 3,080 | 919 | 557 | 382 | 235 |
| 21 | 101x101 | L | 7,456 | 2,232 | 1,352 | 929 | 572 |
| M | 5,712 | 1,708 | 1,035 | 711 | 438 |
| Q | 4,096 | 1,224 | 742 | 509 | 314 |
| H | 3,248 | 969 | 587 | 403 | 248 |
| 22 | 105x105 | L | 8,048 | 2,409 | 1,460 | 1,003 | 618 |
| M | 6,256 | 1,872 | 1,134 | 779 | 480 |
| Q | 4,544 | 1,358 | 823 | 565 | 348 |
| H | 3,536 | 1,056 | 640 | 439 | 270 |
| 23 | 109x109 | L | 8,752 | 2,620 | 1,588 | 1,091 | 672 |
| M | 6,880 | 2,059 | 1,248 | 857 | 528 |
| Q | 4,912 | 1,468 | 890 | 611 | 376 |
| H | 3,712 | 1,108 | 672 | 461 | 284 |
| 24 | 113x113 | L | 9,392 | 2,812 | 1,704 | 1,171 | 721 |
| M | 7,312 | 2,188 | 1,326 | 911 | 561 |
| Q | 5,312 | 1,588 | 963 | 661 | 407 |
| H | 4,112 | 1,228 | 744 | 511 | 315 |
| 25 | 117x117 | L | 10,208 | 3,057 | 1,853 | 1,273 | 784 |
| M | 8,000 | 2,395 | 1,451 | 997 | 614 |
| Q | 5,744 | 1,718 | 1,041 | 715 | 440 |
| H | 4,304 | 1,286 | 779 | 535 | 330 |
| 26 | 121x121 | L | 10,960 | 3,283 | 1,990 | 1,367 | 842 |
| M | 8,496 | 2,544 | 1,542 | 1,059 | 652 |
| Q | 6,032 | 1,804 | 1,094 | 751 | 462 |
| H | 4,768 | 1,425 | 864 | 593 | 365 |
| 27 | 125x125 | L | 11,744 | 3,514 | 2,132 | 1,465 | 902 |
| M | 9,024 | 2,701 | 1,637 | 1,125 | 692 |
| Q | 6,464 | 1,933 | 1,172 | 805 | 496 |
| H | 5,024 | 1,501 | 910 | 625 | 385 |
| 28 | 129x129 | L | 12,248 | 3,669 | 2,223 | 1,528 | 940 |
| M | 9,544 | 2,857 | 1,732 | 1,190 | 732 |
| Q | 6,968 | 2,085 | 1,263 | 868 | 534 |
| H | 5,288 | 1,581 | 958 | 658 | 405 |
| 29 | 133x133 | L | 13,048 | 3,909 | 2,369 | 1,628 | 1,002 |
| M | 10,136 | 3,035 | 1,839 | 1,264 | 778 |
| Q | 7,288 | 2,181 | 1,322 | 908 | 559 |
| H | 5,608 | 1,677 | 1,016 | 698 | 430 |
| 30 | 137x137 | L | 13,880 | 4,158 | 2,520 | 1,732 | 1,066 |
| M | 10,984 | 3,289 | 1,994 | 1,370 | 843 |
| Q | 7,880 | 2,358 | 1,429 | 982 | 604 |
| H | 5,960 | 1,782 | 1,080 | 742 | 457 |
| 31 | 141x141 | L | 14,744 | 4,417 | 2,677 | 1,840 | 1132 |
| M | 11,640 | 3,486 | 2,113 | 1,452 | 894 |
| Q | 8,264 | 2,473 | 1,499 | 1,030 | 634 |
| H | 6,344 | 1,897 | 1,150 | 790 | 486 |
| 32 | 145x145 | L | 15,640 | 4,686 | 2,840 | 1,952 | 1,201 |
| M | 12,328 | 3,693 | 2,238 | 1,538 | 947 |
| Q | 8,920 | 2,670 | 1,618 | 1,112 | 684 |
| H | 6,760 | 2,022 | 1,226 | 842 | 518 |
| 33 | 149x149 | L | 16,568 | 4,965 | 3,009 | 2,068 | 1,273 |
| M | 13,048 | 3,909 | 2,369 | 1,628 | 1,002 |
| Q | 9,368 | 2,805 | 1,700 | 1,168 | 719 |
| H | 7,208 | 2,157 | 1,307 | 898 | 553 |
| 34 | 153x153 | L | 17,528 | 5,253 | 3,183 | 2,188 | 1,347 |
| M | 13,800 | 4,134 | 2,506 | 1,722 | 1,060 |
| Q | 9,848 | 2,949 | 1,787 | 1,228 | 756 |
| H | 7,688 | 2,301 | 1,394 | 958 | 590 |
| 35 | 157x157 | L | 18,448 | 5,529 | 3,351 | 2,303 | 1,417 |
| M | 14,496 | 4,343 | 2,632 | 1,809 | 1,113 |
| Q | 10,288 | 3,081 | 1,867 | 1,283 | 790 |
| H | 7,888 | 2,361 | 1,431 | 983 | 605 |
| 36 | 161x161 | L | 19,472 | 5,836 | 3,537 | 2,431 | 1,496 |
| M | 15,312 | 4,588 | 2,780 | 1,911 | 1,176 |
| Q | 10,832 | 3,244 | 1,966 | 1,351 | 832 |
| H | 8,432 | 2,524 | 1,530 | 1,051 | 647 |
| 37 | 165x165 | L | 20,528 | 6,153 | 3,729 | 2,563 | 1,577 |
| M | 15,936 | 4,775 | 2,894 | 1,989 | 1,224 |
| Q | 11,408 | 3,417 | 2,071 | 1,423 | 876 |
| H | 8,768 | 2,625 | 1,591 | 1,093 | 673 |
| 38 | 169x169 | L | 21,616 | 6,479 | 3,927 | 2,699 | 1,661 |
| M | 16,816 | 5,039 | 3,054 | 2,099 | 1,292 |
| Q | 12,016 | 3,599 | 2,181 | 1,499 | 923 |
| H | 9,136 | 2,735 | 1,658 | 1,139 | 701 |
| 39 | 173x173 | L | 22,496 | 6,743 | 4,087 | 2,809 | 1,729 |
| M | 17,728 | 5,313 | 3,220 | 2,213 | 1,362 |
| Q | 12,656 | 3,791 | 2,298 | 1,579 | 972 |
| H | 9,776 | 2,927 | 1,774 | 1,219 | 750 |
| 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 |

**Sửa lỗi Từ Mã và Thông Tin Khối**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Version and EC Level** | **Total Number of Data Codewords for this Version and EC Level** | **EC Codewords Per Block** | **Number of Blocks in Group 1** | **Number of Data Codewords in Each of Group 1's Blocks** | **Number of Blocks in Group 2** | **Number of Data Codewords in Each of Group 2's Blocks** | **Total Data Codewords** |
| 1-L | 19 | 7 | 1 | 19 |  |  | (19×1) = 19 |
| 1-M | 16 | 10 | 1 | 16 |  |  | (16×1) = 16 |
| 1-Q | 13 | 13 | 1 | 13 |  |  | (13×1) = 13 |
| 1-H | 9 | 17 | 1 | 9 |  |  | (9×1) = 9 |
| 2-L | 34 | 10 | 1 | 34 |  |  | (34×1) = 34 |
| 2-M | 28 | 16 | 1 | 28 |  |  | (28×1) = 28 |
| 2-Q | 22 | 22 | 1 | 22 |  |  | (22×1) = 22 |
| 2-H | 16 | 28 | 1 | 16 |  |  | (16×1) = 16 |
| 3-L | 55 | 15 | 1 | 55 |  |  | (55×1) = 55 |
| 3-M | 44 | 26 | 1 | 44 |  |  | (44×1) = 44 |
| 3-Q | 34 | 18 | 2 | 17 |  |  | (17×2) = 34 |
| 3-H | 26 | 22 | 2 | 13 |  |  | (13×2) = 26 |
| 4-L | 80 | 20 | 1 | 80 |  |  | (80×1) = 80 |
| 4-M | 64 | 18 | 2 | 32 |  |  | (32×2) = 64 |
| 4-Q | 48 | 26 | 2 | 24 |  |  | (24×2) = 48 |
| 4-H | 36 | 16 | 4 | 9 |  |  | (9×4) = 36 |
| 5-L | 108 | 26 | 1 | 108 |  |  | (108×1) = 108 |
| 5-M | 86 | 24 | 2 | 43 |  |  | (43×2) = 86 |
| 5-Q | 62 | 18 | 2 | 15 | 2 | 16 | (15×2) + (16×2) = 62 |
| 5-H | 46 | 22 | 2 | 11 | 2 | 12 | (11×2) + (12×2) = 46 |
| 6-L | 136 | 18 | 2 | 68 |  |  | (68×2) = 136 |
| 6-M | 108 | 16 | 4 | 27 |  |  | (27×4) = 108 |
| 6-Q | 76 | 24 | 4 | 19 |  |  | (19×4) = 76 |
| 6-H | 60 | 28 | 4 | 15 |  |  | (15×4) = 60 |
| 7-L | 156 | 20 | 2 | 78 |  |  | (78×2) = 156 |
| 7-M | 124 | 18 | 4 | 31 |  |  | (31×4) = 124 |
| 7-Q | 88 | 18 | 2 | 14 | 4 | 15 | (14×2) + (15×4) = 88 |
| 7-H | 66 | 26 | 4 | 13 | 1 | 14 | (13×4) + (14×1) = 66 |
| 8-L | 194 | 24 | 2 | 97 |  |  | (97×2) = 194 |
| 8-M | 154 | 22 | 2 | 38 | 2 | 39 | (38×2) + (39×2) = 154 |
| 8-Q | 110 | 22 | 4 | 18 | 2 | 19 | (18×4) + (19×2) = 110 |
| 8-H | 86 | 26 | 4 | 14 | 2 | 15 | (14×4) + (15×2) = 86 |
| 9-L | 232 | 30 | 2 | 116 |  |  | (116×2) = 232 |
| 9-M | 182 | 22 | 3 | 36 | 2 | 37 | (36×3) + (37×2) = 182 |
| 9-Q | 132 | 20 | 4 | 16 | 4 | 17 | (16×4) + (17×4) = 132 |
| 9-H | 100 | 24 | 4 | 12 | 4 | 13 | (12×4) + (13×4) = 100 |
| 10-L | 274 | 18 | 2 | 68 | 2 | 69 | (68×2) + (69×2) = 274 |
| 10-M | 216 | 26 | 4 | 43 | 1 | 44 | (43×4) + (44×1) = 216 |
| 10-Q | 154 | 24 | 6 | 19 | 2 | 20 | (19×6) + (20×2) = 154 |
| 10-H | 122 | 28 | 6 | 15 | 2 | 16 | (15×6) + (16×2) = 122 |
| 11-L | 324 | 20 | 4 | 81 |  |  | (81×4) = 324 |
| 11-M | 254 | 30 | 1 | 50 | 4 | 51 | (50×1) + (51×4) = 254 |
| 11-Q | 180 | 28 | 4 | 22 | 4 | 23 | (22×4) + (23×4) = 180 |
| 11-H | 140 | 24 | 3 | 12 | 8 | 13 | (12×3) + (13×8) = 140 |
| 12-L | 370 | 24 | 2 | 92 | 2 | 93 | (92×2) + (93×2) = 370 |
| 12-M | 290 | 22 | 6 | 36 | 2 | 37 | (36×6) + (37×2) = 290 |
| 12-Q | 206 | 26 | 4 | 20 | 6 | 21 | (20×4) + (21×6) = 206 |
| 12-H | 158 | 28 | 7 | 14 | 4 | 15 | (14×7) + (15×4) = 158 |
| 13-L | 428 | 26 | 4 | 107 |  |  | (107×4) = 428 |
| 13-M | 334 | 22 | 8 | 37 | 1 | 38 | (37×8) + (38×1) = 334 |
| 13-Q | 244 | 24 | 8 | 20 | 4 | 21 | (20×8) + (21×4) = 244 |
| 13-H | 180 | 22 | 12 | 11 | 4 | 12 | (11×12) + (12×4) = 180 |
| 14-L | 461 | 30 | 3 | 115 | 1 | 116 | (115×3) + (116×1) = 461 |
| 14-M | 365 | 24 | 4 | 40 | 5 | 41 | (40×4) + (41×5) = 365 |
| 14-Q | 261 | 20 | 11 | 16 | 5 | 17 | (16×11) + (17×5) = 261 |
| 14-H | 197 | 24 | 11 | 12 | 5 | 13 | (12×11) + (13×5) = 197 |
| 15-L | 523 | 22 | 5 | 87 | 1 | 88 | (87×5) + (88×1) = 523 |
| 15-M | 415 | 24 | 5 | 41 | 5 | 42 | (41×5) + (42×5) = 415 |
| 15-Q | 295 | 30 | 5 | 24 | 7 | 25 | (24×5) + (25×7) = 295 |
| 15-H | 223 | 24 | 11 | 12 | 7 | 13 | (12×11) + (13×7) = 223 |
| 16-L | 589 | 24 | 5 | 98 | 1 | 99 | (98×5) + (99×1) = 589 |
| 16-M | 453 | 28 | 7 | 45 | 3 | 46 | (45×7) + (46×3) = 453 |
| 16-Q | 325 | 24 | 15 | 19 | 2 | 20 | (19×15) + (20×2) = 325 |
| 16-H | 253 | 30 | 3 | 15 | 13 | 16 | (15×3) + (16×13) = 253 |
| 17-L | 647 | 28 | 1 | 107 | 5 | 108 | (107×1) + (108×5) = 647 |
| 17-M | 507 | 28 | 10 | 46 | 1 | 47 | (46×10) + (47×1) = 507 |
| 17-Q | 367 | 28 | 1 | 22 | 15 | 23 | (22×1) + (23×15) = 367 |
| 17-H | 283 | 28 | 2 | 14 | 17 | 15 | (14×2) + (15×17) = 283 |
| 18-L | 721 | 30 | 5 | 120 | 1 | 121 | (120×5) + (121×1) = 721 |
| 18-M | 563 | 26 | 9 | 43 | 4 | 44 | (43×9) + (44×4) = 563 |
| 18-Q | 397 | 28 | 17 | 22 | 1 | 23 | (22×17) + (23×1) = 397 |
| 18-H | 313 | 28 | 2 | 14 | 19 | 15 | (14×2) + (15×19) = 313 |
| 19-L | 795 | 28 | 3 | 113 | 4 | 114 | (113×3) + (114×4) = 795 |
| 19-M | 627 | 26 | 3 | 44 | 11 | 45 | (44×3) + (45×11) = 627 |
| 19-Q | 445 | 26 | 17 | 21 | 4 | 22 | (21×17) + (22×4) = 445 |
| 19-H | 341 | 26 | 9 | 13 | 16 | 14 | (13×9) + (14×16) = 341 |
| 20-L | 861 | 28 | 3 | 107 | 5 | 108 | (107×3) + (108×5) = 861 |
| 20-M | 669 | 26 | 3 | 41 | 13 | 42 | (41×3) + (42×13) = 669 |
| 20-Q | 485 | 30 | 15 | 24 | 5 | 25 | (24×15) + (25×5) = 485 |
| 20-H | 385 | 28 | 15 | 15 | 10 | 16 | (15×15) + (16×10) = 385 |
| 21-L | 932 | 28 | 4 | 116 | 4 | 117 | (116×4) + (117×4) = 932 |
| 21-M | 714 | 26 | 17 | 42 |  |  | (42×17) = 714 |
| 21-Q | 512 | 28 | 17 | 22 | 6 | 23 | (22×17) + (23×6) = 512 |
| 21-H | 406 | 30 | 19 | 16 | 6 | 17 | (16×19) + (17×6) = 406 |
| 22-L | 1006 | 28 | 2 | 111 | 7 | 112 | (111×2) + (112×7) = 1006 |
| 22-M | 782 | 28 | 17 | 46 |  |  | (46×17) = 782 |
| 22-Q | 568 | 30 | 7 | 24 | 16 | 25 | (24×7) + (25×16) = 568 |
| 22-H | 442 | 24 | 34 | 13 |  |  | (13×34) = 442 |
| 23-L | 1094 | 30 | 4 | 121 | 5 | 122 | (121×4) + (122×5) = 1094 |
| 23-M | 860 | 28 | 4 | 47 | 14 | 48 | (47×4) + (48×14) = 860 |
| 23-Q | 614 | 30 | 11 | 24 | 14 | 25 | (24×11) + (25×14) = 614 |
| 23-H | 464 | 30 | 16 | 15 | 14 | 16 | (15×16) + (16×14) = 464 |
| 24-L | 1174 | 30 | 6 | 117 | 4 | 118 | (117×6) + (118×4) = 1174 |
| 24-M | 914 | 28 | 6 | 45 | 14 | 46 | (45×6) + (46×14) = 914 |
| 24-Q | 664 | 30 | 11 | 24 | 16 | 25 | (24×11) + (25×16) = 664 |
| 24-H | 514 | 30 | 30 | 16 | 2 | 17 | (16×30) + (17×2) = 514 |
| 25-L | 1276 | 26 | 8 | 106 | 4 | 107 | (106×8) + (107×4) = 1276 |
| 25-M | 1000 | 28 | 8 | 47 | 13 | 48 | (47×8) + (48×13) = 1000 |
| 25-Q | 718 | 30 | 7 | 24 | 22 | 25 | (24×7) + (25×22) = 718 |
| 25-H | 538 | 30 | 22 | 15 | 13 | 16 | (15×22) + (16×13) = 538 |
| 26-L | 1370 | 28 | 10 | 114 | 2 | 115 | (114×10) + (115×2) = 1370 |
| 26-M | 1062 | 28 | 19 | 46 | 4 | 47 | (46×19) + (47×4) = 1062 |
| 26-Q | 754 | 28 | 28 | 22 | 6 | 23 | (22×28) + (23×6) = 754 |
| 26-H | 596 | 30 | 33 | 16 | 4 | 17 | (16×33) + (17×4) = 596 |
| 27-L | 1468 | 30 | 8 | 122 | 4 | 123 | (122×8) + (123×4) = 1468 |
| 27-M | 1128 | 28 | 22 | 45 | 3 | 46 | (45×22) + (46×3) = 1128 |
| 27-Q | 808 | 30 | 8 | 23 | 26 | 24 | (23×8) + (24×26) = 808 |
| 27-H | 628 | 30 | 12 | 15 | 28 | 16 | (15×12) + (16×28) = 628 |
| 28-L | 1531 | 30 | 3 | 117 | 10 | 118 | (117×3) + (118×10) = 1531 |
| 28-M | 1193 | 28 | 3 | 45 | 23 | 46 | (45×3) + (46×23) = 1193 |
| 28-Q | 871 | 30 | 4 | 24 | 31 | 25 | (24×4) + (25×31) = 871 |
| 28-H | 661 | 30 | 11 | 15 | 31 | 16 | (15×11) + (16×31) = 661 |
| 29-L | 1631 | 30 | 7 | 116 | 7 | 117 | (116×7) + (117×7) = 1631 |
| 29-M | 1267 | 28 | 21 | 45 | 7 | 46 | (45×21) + (46×7) = 1267 |
| 29-Q | 911 | 30 | 1 | 23 | 37 | 24 | (23×1) + (24×37) = 911 |
| 29-H | 701 | 30 | 19 | 15 | 26 | 16 | (15×19) + (16×26) = 701 |
| 30-L | 1735 | 30 | 5 | 115 | 10 | 116 | (115×5) + (116×10) = 1735 |
| 30-M | 1373 | 28 | 19 | 47 | 10 | 48 | (47×19) + (48×10) = 1373 |
| 30-Q | 985 | 30 | 15 | 24 | 25 | 25 | (24×15) + (25×25) = 985 |
| 30-H | 745 | 30 | 23 | 15 | 25 | 16 | (15×23) + (16×25) = 745 |
| 31-L | 1843 | 30 | 13 | 115 | 3 | 116 | (115×13) + (116×3) = 1843 |
| 31-M | 1455 | 28 | 2 | 46 | 29 | 47 | (46×2) + (47×29) = 1455 |
| 31-Q | 1033 | 30 | 42 | 24 | 1 | 25 | (24×42) + (25×1) = 1033 |
| 31-H | 793 | 30 | 23 | 15 | 28 | 16 | (15×23) + (16×28) = 793 |
| 32-L | 1955 | 30 | 17 | 115 |  |  | (115×17) = 1955 |
| 32-M | 1541 | 28 | 10 | 46 | 23 | 47 | (46×10) + (47×23) = 1541 |
| 32-Q | 1115 | 30 | 10 | 24 | 35 | 25 | (24×10) + (25×35) = 1115 |
| 32-H | 845 | 30 | 19 | 15 | 35 | 16 | (15×19) + (16×35) = 845 |
| 33-L | 2071 | 30 | 17 | 115 | 1 | 116 | (115×17) + (116×1) = 2071 |
| 33-M | 1631 | 28 | 14 | 46 | 21 | 47 | (46×14) + (47×21) = 1631 |
| 33-Q | 1171 | 30 | 29 | 24 | 19 | 25 | (24×29) + (25×19) = 1171 |
| 33-H | 901 | 30 | 11 | 15 | 46 | 16 | (15×11) + (16×46) = 901 |
| 34-L | 2191 | 30 | 13 | 115 | 6 | 116 | (115×13) + (116×6) = 2191 |
| 34-M | 1725 | 28 | 14 | 46 | 23 | 47 | (46×14) + (47×23) = 1725 |
| 34-Q | 1231 | 30 | 44 | 24 | 7 | 25 | (24×44) + (25×7) = 1231 |
| 34-H | 961 | 30 | 59 | 16 | 1 | 17 | (16×59) + (17×1) = 961 |
| 35-L | 2306 | 30 | 12 | 121 | 7 | 122 | (121×12) + (122×7) = 2306 |
| 35-M | 1812 | 28 | 12 | 47 | 26 | 48 | (47×12) + (48×26) = 1812 |
| 35-Q | 1286 | 30 | 39 | 24 | 14 | 25 | (24×39) + (25×14) = 1286 |
| 35-H | 986 | 30 | 22 | 15 | 41 | 16 | (15×22) + (16×41) = 986 |
| 36-L | 2434 | 30 | 6 | 121 | 14 | 122 | (121×6) + (122×14) = 2434 |
| 36-M | 1914 | 28 | 6 | 47 | 34 | 48 | (47×6) + (48×34) = 1914 |
| 36-Q | 1354 | 30 | 46 | 24 | 10 | 25 | (24×46) + (25×10) = 1354 |
| 36-H | 1054 | 30 | 2 | 15 | 64 | 16 | (15×2) + (16×64) = 1054 |
| 37-L | 2566 | 30 | 17 | 122 | 4 | 123 | (122×17) + (123×4) = 2566 |
| 37-M | 1992 | 28 | 29 | 46 | 14 | 47 | (46×29) + (47×14) = 1992 |
| 37-Q | 1426 | 30 | 49 | 24 | 10 | 25 | (24×49) + (25×10) = 1426 |
| 37-H | 1096 | 30 | 24 | 15 | 46 | 16 | (15×24) + (16×46) = 1096 |
| 38-L | 2702 | 30 | 4 | 122 | 18 | 123 | (122×4) + (123×18) = 2702 |
| 38-M | 2102 | 28 | 13 | 46 | 32 | 47 | (46×13) + (47×32) = 2102 |
| 38-Q | 1502 | 30 | 48 | 24 | 14 | 25 | (24×48) + (25×14) = 1502 |
| 38-H | 1142 | 30 | 42 | 15 | 32 | 16 | (15×42) + (16×32) = 1142 |
| 39-L | 2812 | 30 | 20 | 117 | 4 | 118 | (117×20) + (118×4) = 2812 |
| 39-M | 2216 | 28 | 40 | 47 | 7 | 48 | (47×40) + (48×7) = 2216 |
| 39-Q | 1582 | 30 | 43 | 24 | 22 | 25 | (24×43) + (25×22) = 1582 |
| 39-H | 1222 | 30 | 10 | 15 | 67 | 16 | (15×10) + (16×67) = 1222 |
| 40-L | 2956 | 30 | 19 | 118 | 6 | 119 | (118×19) + (119×6) = 2956 |
| 40-M | 2334 | 28 | 18 | 47 | 31 | 48 | (47×18) + (48×31) = 2334 |
| 40-Q | 1666 | 30 | 34 | 24 | 34 | 25 | (24×34) + (25×34) = 1666 |
| 40-H | 1276 | 30 | 20 | 15 | 61 | 16 | (15×20) + (16×61) = 1276 |

**QR Code Log Antilog Table for Galois Field 256**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Exponent of α** | **Integer** |  | **Integer** | **Exponent of α** |
| 0 | 1 |  |  |  |
| 1 | 2 |  | 1 | 0 |
| 2 | 4 |  | 2 | 1 |
| 3 | 8 |  | 3 | 25 |
| 4 | 16 |  | 4 | 2 |
| 5 | 32 |  | 5 | 50 |
| 6 | 64 |  | 6 | 26 |
| 7 | 128 |  | 7 | 198 |
| 8 | 29 |  | 8 | 3 |
| 9 | 58 |  | 9 | 223 |
| 10 | 116 |  | 10 | 51 |
| 11 | 232 |  | 11 | 238 |
| 12 | 205 |  | 12 | 27 |
| 13 | 135 |  | 13 | 104 |
| 14 | 19 |  | 14 | 199 |
| 15 | 38 |  | 15 | 75 |
| 16 | 76 |  | 16 | 4 |
| 17 | 152 |  | 17 | 100 |
| 18 | 45 |  | 18 | 224 |
| 19 | 90 |  | 19 | 14 |
| 20 | 180 |  | 20 | 52 |
| 21 | 117 |  | 21 | 141 |
| 22 | 234 |  | 22 | 239 |
| 23 | 201 |  | 23 | 129 |
| 24 | 143 |  | 24 | 28 |
| 25 | 3 |  | 25 | 193 |
| 26 | 6 |  | 26 | 105 |
| 27 | 12 |  | 27 | 248 |
| 28 | 24 |  | 28 | 200 |
| 29 | 48 |  | 29 | 8 |
| 30 | 96 |  | 30 | 76 |
| 31 | 192 |  | 31 | 113 |
| 32 | 157 |  | 32 | 5 |
| 33 | 39 |  | 33 | 138 |
| 34 | 78 |  | 34 | 101 |
| 35 | 156 |  | 35 | 47 |
| 36 | 37 |  | 36 | 225 |
| 37 | 74 |  | 37 | 36 |
| 38 | 148 |  | 38 | 15 |
| 39 | 53 |  | 39 | 33 |
| 40 | 106 |  | 40 | 53 |
| 41 | 212 |  | 41 | 147 |
| 42 | 181 |  | 42 | 142 |
| 43 | 119 |  | 43 | 218 |
| 44 | 238 |  | 44 | 240 |
| 45 | 193 |  | 45 | 18 |
| 46 | 159 |  | 46 | 130 |
| 47 | 35 |  | 47 | 69 |
| 48 | 70 |  | 48 | 29 |
| 49 | 140 |  | 49 | 181 |
| 50 | 5 |  | 50 | 194 |
| 51 | 10 |  | 51 | 125 |
| 52 | 20 |  | 52 | 106 |
| 53 | 40 |  | 53 | 39 |
| 54 | 80 |  | 54 | 249 |
| 55 | 160 |  | 55 | 185 |
| 56 | 93 |  | 56 | 201 |
| 57 | 186 |  | 57 | 154 |
| 58 | 105 |  | 58 | 9 |
| 59 | 210 |  | 59 | 120 |
| 60 | 185 |  | 60 | 77 |
| 61 | 111 |  | 61 | 228 |
| 62 | 222 |  | 62 | 114 |
| 63 | 161 |  | 63 | 166 |
| 64 | 95 |  | 64 | 6 |
| 65 | 190 |  | 65 | 191 |
| 66 | 97 |  | 66 | 139 |
| 67 | 194 |  | 67 | 98 |
| 68 | 153 |  | 68 | 102 |
| 69 | 47 |  | 69 | 221 |
| 70 | 94 |  | 70 | 48 |
| 71 | 188 |  | 71 | 253 |
| 72 | 101 |  | 72 | 226 |
| 73 | 202 |  | 73 | 152 |
| 74 | 137 |  | 74 | 37 |
| 75 | 15 |  | 75 | 179 |
| 76 | 30 |  | 76 | 16 |
| 77 | 60 |  | 77 | 145 |
| 78 | 120 |  | 78 | 34 |
| 79 | 240 |  | 79 | 136 |
| 80 | 253 |  | 80 | 54 |
| 81 | 231 |  | 81 | 208 |
| 82 | 211 |  | 82 | 148 |
| 83 | 187 |  | 83 | 206 |
| 84 | 107 |  | 84 | 143 |
| 85 | 214 |  | 85 | 150 |
| 86 | 177 |  | 86 | 219 |
| 87 | 127 |  | 87 | 189 |
| 88 | 254 |  | 88 | 241 |
| 89 | 225 |  | 89 | 210 |
| 90 | 223 |  | 90 | 19 |
| 91 | 163 |  | 91 | 92 |
| 92 | 91 |  | 92 | 131 |
| 93 | 182 |  | 93 | 56 |
| 94 | 113 |  | 94 | 70 |
| 95 | 226 |  | 95 | 64 |
| 96 | 217 |  | 96 | 30 |
| 97 | 175 |  | 97 | 66 |
| 98 | 67 |  | 98 | 182 |
| 99 | 134 |  | 99 | 163 |
| 100 | 17 |  | 100 | 195 |
| 101 | 34 |  | 101 | 72 |
| 102 | 68 |  | 102 | 126 |
| 103 | 136 |  | 103 | 110 |
| 104 | 13 |  | 104 | 107 |
| 105 | 26 |  | 105 | 58 |
| 106 | 52 |  | 106 | 40 |
| 107 | 104 |  | 107 | 84 |
| 108 | 208 |  | 108 | 250 |
| 109 | 189 |  | 109 | 133 |
| 110 | 103 |  | 110 | 186 |
| 111 | 206 |  | 111 | 61 |
| 112 | 129 |  | 112 | 202 |
| 113 | 31 |  | 113 | 94 |
| 114 | 62 |  | 114 | 155 |
| 115 | 124 |  | 115 | 159 |
| 116 | 248 |  | 116 | 10 |
| 117 | 237 |  | 117 | 21 |
| 118 | 199 |  | 118 | 121 |
| 119 | 147 |  | 119 | 43 |
| 120 | 59 |  | 120 | 78 |
| 121 | 118 |  | 121 | 212 |
| 122 | 236 |  | 122 | 229 |
| 123 | 197 |  | 123 | 172 |
| 124 | 151 |  | 124 | 115 |
| 125 | 51 |  | 125 | 243 |
| 126 | 102 |  | 126 | 167 |
| 127 | 204 |  | 127 | 87 |
| 128 | 133 |  | 128 | 7 |
| 129 | 23 |  | 129 | 112 |
| 130 | 46 |  | 130 | 192 |
| 131 | 92 |  | 131 | 247 |
| 132 | 184 |  | 132 | 140 |
| 133 | 109 |  | 133 | 128 |
| 134 | 218 |  | 134 | 99 |
| 135 | 169 |  | 135 | 13 |
| 136 | 79 |  | 136 | 103 |
| 137 | 158 |  | 137 | 74 |
| 138 | 33 |  | 138 | 222 |
| 139 | 66 |  | 139 | 237 |
| 140 | 132 |  | 140 | 49 |
| 141 | 21 |  | 141 | 197 |
| 142 | 42 |  | 142 | 254 |
| 143 | 84 |  | 143 | 24 |
| 144 | 168 |  | 144 | 227 |
| 145 | 77 |  | 145 | 165 |
| 146 | 154 |  | 146 | 153 |
| 147 | 41 |  | 147 | 119 |
| 148 | 82 |  | 148 | 38 |
| 149 | 164 |  | 149 | 184 |
| 150 | 85 |  | 150 | 180 |
| 151 | 170 |  | 151 | 124 |
| 152 | 73 |  | 152 | 17 |
| 153 | 146 |  | 153 | 68 |
| 154 | 57 |  | 154 | 146 |
| 155 | 114 |  | 155 | 217 |
| 156 | 228 |  | 156 | 35 |
| 157 | 213 |  | 157 | 32 |
| 158 | 183 |  | 158 | 137 |
| 159 | 115 |  | 159 | 46 |
| 160 | 230 |  | 160 | 55 |
| 161 | 209 |  | 161 | 63 |
| 162 | 191 |  | 162 | 209 |
| 163 | 99 |  | 163 | 91 |
| 164 | 198 |  | 164 | 149 |
| 165 | 145 |  | 165 | 188 |
| 166 | 63 |  | 166 | 207 |
| 167 | 126 |  | 167 | 205 |
| 168 | 252 |  | 168 | 144 |
| 169 | 229 |  | 169 | 135 |
| 170 | 215 |  | 170 | 151 |
| 171 | 179 |  | 171 | 178 |
| 172 | 123 |  | 172 | 220 |
| 173 | 246 |  | 173 | 252 |
| 174 | 241 |  | 174 | 190 |
| 175 | 255 |  | 175 | 97 |
| 176 | 227 |  | 176 | 242 |
| 177 | 219 |  | 177 | 86 |
| 178 | 171 |  | 178 | 211 |
| 179 | 75 |  | 179 | 171 |
| 180 | 150 |  | 180 | 20 |
| 181 | 49 |  | 181 | 42 |
| 182 | 98 |  | 182 | 93 |
| 183 | 196 |  | 183 | 158 |
| 184 | 149 |  | 184 | 132 |
| 185 | 55 |  | 185 | 60 |
| 186 | 110 |  | 186 | 57 |
| 187 | 220 |  | 187 | 83 |
| 188 | 165 |  | 188 | 71 |
| 189 | 87 |  | 189 | 109 |
| 190 | 174 |  | 190 | 65 |
| 191 | 65 |  | 191 | 162 |
| 192 | 130 |  | 192 | 31 |
| 193 | 25 |  | 193 | 45 |
| 194 | 50 |  | 194 | 67 |
| 195 | 100 |  | 195 | 216 |
| 196 | 200 |  | 196 | 183 |
| 197 | 141 |  | 197 | 123 |
| 198 | 7 |  | 198 | 164 |
| 199 | 14 |  | 199 | 118 |
| 200 | 28 |  | 200 | 196 |
| 201 | 56 |  | 201 | 23 |
| 202 | 112 |  | 202 | 73 |
| 203 | 224 |  | 203 | 236 |
| 204 | 221 |  | 204 | 127 |
| 205 | 167 |  | 205 | 12 |
| 206 | 83 |  | 206 | 111 |
| 207 | 166 |  | 207 | 246 |
| 208 | 81 |  | 208 | 108 |
| 209 | 162 |  | 209 | 161 |
| 210 | 89 |  | 210 | 59 |
| 211 | 178 |  | 211 | 82 |
| 212 | 121 |  | 212 | 41 |
| 213 | 242 |  | 213 | 157 |
| 214 | 249 |  | 214 | 85 |
| 215 | 239 |  | 215 | 170 |
| 216 | 195 |  | 216 | 251 |
| 217 | 155 |  | 217 | 96 |
| 218 | 43 |  | 218 | 134 |
| 219 | 86 |  | 219 | 177 |
| 220 | 172 |  | 220 | 187 |
| 221 | 69 |  | 221 | 204 |
| 222 | 138 |  | 222 | 62 |
| 223 | 9 |  | 223 | 90 |
| 224 | 18 |  | 224 | 203 |
| 225 | 36 |  | 225 | 89 |
| 226 | 72 |  | 226 | 95 |
| 227 | 144 |  | 227 | 176 |
| 228 | 61 |  | 228 | 156 |
| 229 | 122 |  | 229 | 169 |
| 230 | 244 |  | 230 | 160 |
| 231 | 245 |  | 231 | 81 |
| 232 | 247 |  | 232 | 11 |
| 233 | 243 |  | 233 | 245 |
| 234 | 251 |  | 234 | 22 |
| 235 | 235 |  | 235 | 235 |
| 236 | 203 |  | 236 | 122 |
| 237 | 139 |  | 237 | 117 |
| 238 | 11 |  | 238 | 44 |
| 239 | 22 |  | 239 | 215 |
| 240 | 44 |  | 240 | 79 |
| 241 | 88 |  | 241 | 174 |
| 242 | 176 |  | 242 | 213 |
| 243 | 125 |  | 243 | 233 |
| 244 | 250 |  | 244 | 230 |
| 245 | 233 |  | 245 | 231 |
| 246 | 207 |  | 246 | 173 |
| 247 | 131 |  | 247 | 232 |
| 248 | 27 |  | 248 | 116 |
| 249 | 54 |  | 249 | 214 |
| 250 | 108 |  | 250 | 244 |
| 251 | 216 |  | 251 | 234 |
| 252 | 173 |  | 252 | 168 |
| 253 | 71 |  | 253 | 80 |
| 254 | 142 |  | 254 | 88 |
| 255 | 1 |  | 255 | 175 |

**Cách tạo đa thức sinh**

*Giới thiệu về tạo đa thức sinh*

Dưới đây, bạn sẽ tìm thấy các bước để tạo đa thức tạo cho 13 từ mã sửa lỗi.

Tuy nhiên, vì chúng ta đang làm việc với trường galois GF256, nên chúng ta cần sử dụng các phương pháp đặc biệt để đảm bảo các số của chúng ta không quá lớn hoặc quá nhỏ để nằm trong trường galois.

Khi nhân hai giá trị alpha, thông thường chúng ta cộng hai số mũ với nhau. Nhưng nếu kết quả lớn hơn 255 thì chúng ta có vấn đề.

Ví dụ, hãy xem xét α 251 × α 10 . Nhân hai số này ta có α 261 , nhưng nó quá lớn.

Để ngăn điều đó xảy ra, chúng ta đặt số mũ kết quả thông qua công thức sau:  
(số mũ % 256) + tầng(số mũ / 256)

Sử dụng số mũ của 261 từ ví dụ trên, chúng ta có:  
(261 % 256) + tầng (261/256) = (5) + tầng (1,01953125) = (5 + 1) = 6.

Vậy khi ta nhân α 251 × α 10 thì được kết quả là α 6 .

Một vấn đề khác là kết hợp các thuật ngữ giống nhau. Do tính chất của trường galois, chúng ta không thể cộng các số lại với nhau một cách đơn giản. Chúng ta phải XOR chúng.

Điều này sẽ gây ra một số kết quả không mong muốn. Ví dụ:  
α 201 x 2 + α 199 x 2 = 56x 2 + 14x 2

Nếu chúng ta cộng chúng lại với nhau một cách bình thường thì chúng ta sẽ được 70x 2 . Con số này không lớn hơn 255 nên có vẻ chấp nhận được. Tuy nhiên, nếu chúng ta thực hiện XOR thay vì phép cộng, chúng ta sẽ nhận được:  
56 ⊕ 14 = 54.

Đây là kết quả đúng. Khi cộng các số hạng cùng loại trong quá trình tạo đa thức sinh, bạn phải luôn XOR các số nguyên thay vì cộng chúng.

*Cách tạo đa thức sinh*

Trong mỗi bước tạo đa thức sinh, bạn nhân đa thức với một đa thức. Đa thức đầu tiên mà bạn bắt đầu ở bước đầu tiên luôn là *(α 0 x 1 + α 0 x 0 )* . Đối với mỗi bước nhân, bạn nhân đa thức hiện tại với *(α 0 x 1 + α j x 0 )* trong đó *j* là 1 cho phép nhân đầu tiên, 2 cho phép nhân thứ hai, 3 cho phép nhân thứ ba, v.v.

*Bước nhân số #1*

Chúng ta sẽ nhân (α 0 x 1 + α 0 x 0 ) với (α 0 x 1 + α 1 x 0 )

Sau khi nhân mỗi số hạng ở phần đầu tiên với mỗi số hạng ở phần thứ hai, chúng ta nhận được:  
(α 0 x 1 × α 0 x 1 ) + (α 0 x 0 × α 0 x 1 ) + (α 0 x 1 × α 1 x 0 ) + (α 0 x 0 × α 1 x 0 )

Để nhân, bạn cộng các số mũ lại với nhau như thế này:  
(α (0+0) x (1+1) ) + (α (0+0) x (0+1) ) + (α (0+1) x ( 1+0) ) + (α (0+1) x (0+0) )

Sau khi cộng số mũ ta được kết quả:  
α 0 x 2 + α 0 x 1 + α 1 x 1 + α 1 x 0

Nếu có bất kỳ số mũ nào lớn hơn 255, hãy sửa chúng bằng cách đưa chúng vào công thức  
(số mũ % 256) + tầng (số mũ / 256). Trong trường hợp này, tất cả số mũ đều nhỏ hơn hoặc bằng 255 nên chúng ta có thể tiếp tục.

Bây giờ bạn phải kết hợp các số hạng có cùng số mũ. Trong trường hợp này, có nhiều hơn một số hạng x 1 .

Bây giờ chúng ta sẽ kết hợp các số hạng x 1 . chúng là:  
α 0 x 1 + α 1 x 1

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
1x 1 + 2x 1

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(1 ⊕ 2)x 1 = 3x 1 = α 25 x 1

Dưới đây là đa thức cuối cùng của bước này sau khi kết hợp các số hạng giống nhau. **Nếu chúng ta cần 2 từ mã sửa lỗi thì đây sẽ là đa thức sinh được yêu cầu.**  
α 0 x 2 + α 25 x 1 + α 1 x 0

*Bước nhân số 2*

Sử dụng đa thức thu được ở bước trước, chúng ta sẽ nhân (α 0 x 2 + α 25 x 1 + α 1 x 0 ) với (α 0 x 1 + α 2 x 0 )

Sau khi nhân mỗi số hạng ở phần thứ nhất với mỗi số hạng ở phần thứ hai, chúng ta nhận được:  
(α 0 x 2 × α 0 x 1 ) + (α 25 x 1 × α 0 x 1 ) + (α 1 x 0 × α 0 x 1 ) + (α 0 x 2 × α 2 x 0 ) + (α 25 x 1 × α 2 x 0 ) + (α 1 x 0 × α 2 x 0 )

Để nhân, bạn cộng các số mũ lại với nhau như sau:  
(α (0+0) x (2+1) ) + (α (25+0) x (1+1) ) + (α (1+0) x ( 0+1) ) + (α (0+2) x (2+0) ) + (α (25+2) x (1+0) ) + (α (1+2) x (0+0) )

Sau khi cộng số mũ ta được kết quả:  
α 0 x 3 + α 25 x 2 + α 1 x 1 + α 2 x 2 + α 27 x 1 + α 3 x 0

Nếu có bất kỳ số mũ nào lớn hơn 255, hãy sửa chúng bằng cách đưa chúng vào công thức  
(số mũ % 256) + tầng (số mũ / 256). Trong trường hợp này, tất cả số mũ đều nhỏ hơn hoặc bằng 255 nên chúng ta có thể tiếp tục.

Bây giờ bạn phải kết hợp các số hạng có cùng số mũ. Trong trường hợp này, có nhiều hơn một số hạng x 2 và x 1 .

Bây giờ chúng ta sẽ kết hợp các số hạng x 2 . chúng là:  
α 25 x 2 + α 2 x 2

Chúng ta sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
3x 2 + 4x 2

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(3 ⊕ 4)x 2 = 7x 2 = α 198 x 2

Bây giờ chúng ta sẽ kết hợp các số hạng x 1 . chúng là:  
α 1 x 1 + α 27 x 1

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
2x 1 + 12x 1

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(2 ⊕ 12)x 1 = 14x 1 = α 199 x 1

Dưới đây là đa thức cuối cùng của bước này sau khi kết hợp các số hạng giống nhau. **Nếu chúng ta cần 3 từ mã sửa lỗi thì đây sẽ là đa thức sinh được yêu cầu.**  
α 0 x 3 + α 198 x 2 + α 199 x 1 + α 3 x 0

*Bước nhân số 3*

Sử dụng đa thức thu được ở bước trước, chúng ta sẽ nhân (α 0 x 3 + α 198 x 2 + α 199 x 1 + α 3 x 0 ) với (α 0 x 1 + α 3 x 0 )

Sau khi nhân mỗi số hạng ở phần đầu tiên với mỗi số hạng ở phần thứ hai, chúng ta nhận được:  
(α 0 x 3 × α 0 x 1 ) + (α 198 x 2 × α 0 x 1 ) + (α 199 x 1 × α 0 x 1 ) + (α 3 x 0 × α 0 x 1 ) + (α 0 x 3 × α 3 x 0 ) + (α 198 x 2 × α 3 x 0 ) + (α 199 x 1 × α 3 x 0 ) + (α 3 x 0 × α 3 x 0 )

Để nhân, bạn cộng các số mũ lại với nhau như thế này:  
(α (0+0) x (3+1) ) + (α (198+0) x (2+1) ) + (α (199+0) x ( 1+1) ) + (α (3+0) x (0+1) ) + (α (0+3) x (3+0) ) + (α (198+3) x (2+0) ) + (α (199+3) x (1+0) ) + (α (3+3) x (0+0) )

Sau khi cộng số mũ ta được kết quả:  
α 0 x 4 + α 198 x 3 + α 199 x 2 + α 3 x 1 + α 3 x 3 + α 201 x 2 + α 202 x 1 + α 6 x 0

Nếu có bất kỳ số mũ nào lớn hơn 255, hãy sửa chúng bằng cách đưa chúng vào công thức  
(số mũ % 256) + tầng (số mũ / 256). Trong trường hợp này, tất cả số mũ đều nhỏ hơn hoặc bằng 255 nên chúng ta có thể tiếp tục.

Bây giờ bạn phải kết hợp các số hạng có cùng số mũ. Trong trường hợp này, có nhiều hơn một số hạng x 3 , x 2 và x 1 .

Bây giờ chúng ta sẽ kết hợp các số hạng x 3 . chúng là:  
α 198 x 3 + α 3 x 3

Chúng ta sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
7x 3 + 8x 3

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(7 ⊕ 8)x 3 = 15x 3 = α 75 x 3

Bây giờ chúng ta sẽ kết hợp các số hạng x 2 . chúng là:  
α 199 x 2 + α 201 x 2

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
14x 2 + 56x 2

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(14 ⊕ 56)x 2 = 54x 2 = α 249 x 2

Bây giờ chúng ta sẽ kết hợp các số hạng x 1 . chúng là:  
α 3 x 1 + α 202 x 1

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
8x 1 + 112x 1

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(8 ⊕ 112)x 1 = 120x 1 = α 78 x 1

Dưới đây là đa thức cuối cùng của bước này sau khi kết hợp các số hạng giống nhau. **Nếu chúng ta cần 4 từ mã sửa lỗi thì đây sẽ là đa thức sinh được yêu cầu.**  
α 0 x 4 + α 75 x 3 + α 249 x 2 + α 78 x 1 + α 6 x 0

*Bước nhân số 4*

Sử dụng đa thức thu được ở bước trước, chúng ta sẽ nhân (α 0 x 4 + α 75 x 3 + α 249 x 2 + α 78 x 1 + α 6 x 0 ) với (α 0 x 1 + α 4 x 0 )

Sau khi nhân mỗi số hạng ở phần đầu tiên với mỗi số hạng ở phần thứ hai, chúng ta nhận được:  
(α 0 x 4 × α 0 x 1 ) + (α 75 x 3 × α 0 x 1 ) + (α 249 x 2 × α 0 x 1 ) + (α 78 x 1 × α 0 x 1 ) + (α 6 x 0 × α 0 x 1 ) + (α 0 x 4 × α 4 x 0 ) + (α 75 x 3 × α 4 x 0 ) + (α 249 x 2 × α 4 x 0 ) + (α 78 x 1 × α 4 x 0 ) + (α 6 x 0 × α 4 x 0 )

Để nhân, bạn cộng các số mũ lại với nhau như thế này:  
(α (0+0) x (4+1) ) + (α (75+0) x (3+1) ) + (α (249+0) x ( 2+1) ) + (α (78+0) x (1+1) ) + (α (6+0) x (0+1) ) + (α (0+4) x (4+0) ) + (α (75+4) x (3+0) ) + (α (249+4) x (2+0) ) + (α (78+4) x (1+0) ) + (α (6 +4) x (0+0) )

Sau khi cộng số mũ ta được kết quả:  
α 0 x 5 + α 75 x 4 + α 249 x 3 + α 78 x 2 + α 6 x 1 + α 4 x 4 + α 79 x 3 + α 253 x 2 + α 82 x 1 + α 10 x 0

Nếu có bất kỳ số mũ nào lớn hơn 255, hãy sửa chúng bằng cách đưa chúng vào công thức  
(số mũ % 256) + tầng (số mũ / 256). Trong trường hợp này, tất cả số mũ đều nhỏ hơn hoặc bằng 255 nên chúng ta có thể tiếp tục.

Bây giờ bạn phải kết hợp các số hạng có cùng số mũ. Trong trường hợp này, có nhiều hơn một số hạng x 4 , x 3 , x 2 và x 1 .

Bây giờ chúng ta sẽ kết hợp các số hạng x 4 . chúng là:  
α 75 x 4 + α 4 x 4

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
15x 4 + 16x 4

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(15 ⊕ 16)x 4 = 31x 4 = α 113 x 4

Bây giờ chúng ta sẽ kết hợp các số hạng x 3 . chúng là:  
α 249 x 3 + α 79 x 3

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
54x 3 + 240x 3

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(54 ⊕ 240)x 3 = 198x 3 = α 164 x 3

Bây giờ chúng ta sẽ kết hợp các số hạng x 2 . chúng là:  
α 78 x 2 + α 253 x 2

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
120x 2 + 71x 2

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(120 ⊕ 71)x 2 = 63x 2 = α 166 x 2

Bây giờ chúng ta sẽ kết hợp các số hạng x 1 . chúng là:  
α 6 x 1 + α 82 x 1

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
64x 1 + 211x 1

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(64 ⊕ 211)x 1 = 147x 1 = α 119 x 1

Dưới đây là đa thức cuối cùng của bước này sau khi kết hợp các số hạng giống nhau. **Nếu chúng ta cần 5 từ mã sửa lỗi thì đây sẽ là đa thức sinh được yêu cầu.**  
α 0 x 5 + α 113 x 4 + α 164 x 3 + α 166 x 2 + α 119 x 1 + α 10 x 0

*Bước nhân #5*

Sử dụng đa thức thu được ở bước trước, chúng ta sẽ nhân (α 0 x 5 + α 113 x 4 + α 164 x 3 + α 166 x 2 + α 119 x 1 + α 10 x 0 ) với (α 0 x 1 + α 5 x 0 )

Sau khi nhân mỗi số hạng ở phần thứ nhất với mỗi số hạng ở phần thứ hai, ta được:  
(α 0 x 5 × α 0 x 1 ) + (α 113 x 4 × α 0 x 1 ) + (α 164 x 3 × α 0 x 1 ) + (α 166 x 2 × α 0 x 1 ) + (α 119 x 1 × α 0 x 1 ) + (α 10 x 0 × α 0 x 1 ) + (α 0 x 5 × α 5 x 0 ) + (α 113 x 4 × α 5 x 0 ) + (α 164 x 3 × α 5 x 0 ) + (α 166 x 2 × α 5 x 0 ) + (α 119 x 1 × α 5 x 0 ) + (α 10 x 0 × α 5 x 0 )

Để nhân, bạn cộng các số mũ lại với nhau như thế này:  
(α (0+0) x (5+1) ) + (α (113+0) x (4+1) ) + (α (164+0) x ( 3+1) ) + (α (166+0) x (2+1) ) + (α (119+0) x (1+1) ) + (α (10+0) x (0+1) ) + (α (0+5) x (5+0) ) + (α (113+5) x (4+0) ) + (α (164+5) x (3+0) ) + (α (166 +5) x (2+0) ) + (α (119+5) x (1+0) ) + (α (10+5) x (0+0) )

Sau khi cộng số mũ ta được kết quả:  
α 0 x 6 + α 113 x 5 + α 164 x 4 + α 166 x 3 + α 119 x 2 + α 10 x 1 + α 5 x 5 + α 118 x 4 + α 169 x 3 + α 171 x 2 + α 124 x 1 + α 15 x 0

Nếu có bất kỳ số mũ nào lớn hơn 255, hãy sửa chúng bằng cách đưa chúng vào công thức  
(số mũ % 256) + tầng (số mũ / 256). Trong trường hợp này, tất cả số mũ đều nhỏ hơn hoặc bằng 255 nên chúng ta có thể tiếp tục.

Bây giờ bạn phải kết hợp các số hạng có cùng số mũ. Trong trường hợp này, có nhiều hơn một số hạng x 5 , x 4 , x 3 , x 2 và x 1 .

Bây giờ chúng ta sẽ kết hợp các số hạng x 5 . chúng là:  
α 113 x 5 + α 5 x 5

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
31x 5 + 32x 5

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(31 ⊕ 32)x 5 = 63x 5 = α 166 x 5

Bây giờ chúng ta sẽ kết hợp các số hạng x 4 . chúng là:  
α 164 x 4 + α 118 x 4

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
198x 4 + 199x 4

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(198 ⊕ 199)x 4 = 1x 4 = α 0 x 4

Bây giờ chúng ta sẽ kết hợp các số hạng x 3 . chúng là:  
α 166 x 3 + α 169 x 3

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
63x 3 + 229x 3

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(63 ⊕ 229)x 3 = 218x 3 = α 134 x 3

Bây giờ chúng ta sẽ kết hợp các số hạng x 2 . chúng là:  
α 119 x 2 + α 171 x 2

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
147x 2 + 179x 2

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(147 ⊕ 179)x 2 = 32x 2 = α 5 x 2

Bây giờ chúng ta sẽ kết hợp các số hạng x 1 . chúng là:  
α 10 x 1 + α 124 x 1

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
116x 1 + 151x 1

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(116 ⊕ 151)x 1 = 227x 1 = α 176 x 1

Dưới đây là đa thức cuối cùng của bước này sau khi kết hợp các số hạng giống nhau. **Nếu chúng ta cần 6 từ mã sửa lỗi thì đây sẽ là đa thức sinh được yêu cầu.**  
α 0 x 6 + α 166 x 5 + α 0 x 4 + α 134 x 3 + α 5 x 2 + α 176 x 1 + α 15 x 0

*Bước nhân #6*

Sử dụng đa thức thu được ở bước trước, chúng ta sẽ nhân (α 0 x 6 + α 166 x 5 + α 0 x 4 + α 134 x 3 + α 5 x 2 + α 176 x 1 + α 15 x 0 ) bởi (α 0 x 1 + α 6 x 0 )

Sau khi nhân mỗi số hạng ở phần đầu tiên với mỗi số hạng ở phần thứ hai, chúng ta nhận được:  
(α 0 x 6 × α 0 x 1 ) + (α 166 x 5 × α 0 x 1 ) + (α 0 x 4 × α 0 x 1 ) + (α 134 x 3 × α 0 x 1 ) + (α 5 x 2 × α 0 x 1 ) + (α 176 x 1 × α 0 x 1 ) + (α 15 x 0 × α 0 x 1 ) + (α 0 x 6 × α 6 x 0 ) + (α 166 x 5 × α 6 x 0 ) + (α 0 x 4 × α 6 x 0 ) + (α 134 x 3 × α 6 x 0 ) + (α 5 x 2 × α 6 x 0 ) + (α 176 x 1 × α 6 x 0 ) + (α 15 x 0 × α 6 x 0 )

Để nhân, bạn cộng các số mũ lại với nhau như thế này:  
(α (0+0) x (6+1) ) + (α (166+0) x (5+1) ) + (α (0+0) x ( 4+1) ) + (α (134+0) x (3+1) ) + (α (5+0) x (2+1) ) + (α (176+0) x (1+1) ) + (α (15+0) x (0+1) ) + (α (0+6) x (6+0) ) + (α (166+6) x (5+0) ) + (α (0 +6) x (4+0) ) + (α (134+6) x (3+0) ) + (α (5+6) x (2+0) ) + (α (176+6) x ( 1+0) ) + (α (15+6) x (0+0) )

Sau khi cộng số mũ ta được kết quả:  
α 0 x 7 + α 166 x 6 + α 0 x 5 + α 134 x 4 + α 5 x 3 + α 176 x 2 + α 15 x 1 + α 6 x 6 + α 172 x 5 + α 6 x 4 + α 140 x 3 + α 11 x 2 + α 182 x 1 + α 21 x 0

Nếu có bất kỳ số mũ nào lớn hơn 255, hãy sửa chúng bằng cách đưa chúng vào công thức  
(số mũ % 256) + tầng (số mũ / 256). Trong trường hợp này, tất cả số mũ đều nhỏ hơn hoặc bằng 255 nên chúng ta có thể tiếp tục.

Bây giờ bạn phải kết hợp các số hạng có cùng số mũ. Trong trường hợp này, có nhiều hơn một số hạng x 6 , x 5 , x 4 , x 3 , x 2 và x 1 .

Bây giờ chúng ta sẽ kết hợp các số hạng x 6 . chúng là:  
α 166 x 6 + α 6 x 6

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
63x 6 + 64x 6

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(63 ⊕ 64)x 6 = 127x 6 = α 87 x 6

Bây giờ chúng ta sẽ kết hợp các số hạng x 5 . chúng là:  
α 0 x 5 + α 172 x 5

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
1x 5 + 123x 5

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(1 ⊕ 123)x 5 = 122x 5 = α 229 x 5

Bây giờ chúng ta sẽ kết hợp các số hạng x 4 . chúng là:  
α 134 x 4 + α 6 x 4

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
218x 4 + 64x 4

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(218 ⊕ 64)x4 = 154x4 = α 146 x 4

Bây giờ chúng ta sẽ kết hợp các số hạng x 3 . chúng là:  
α 5 x 3 + α 140 x 3

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
32x 3 + 132x 3

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(32 ⊕ 132)x 3 = 164x 3 = α 149 x 3

Bây giờ chúng ta sẽ kết hợp các số hạng x 2 . chúng là:  
α 176 x 2 + α 11 x 2

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
227x 2 + 232x 2

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(227 ⊕ 232)x 2 = 11x 2 = α 238 x 2

Bây giờ chúng ta sẽ kết hợp các số hạng x 1 . chúng là:  
α 15 x 1 + α 182 x 1

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
38x 1 + 98x 1

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(38 ⊕ 98)x 1 = 68x 1 = α 102 x 1

Dưới đây là đa thức cuối cùng của bước này sau khi kết hợp các số hạng giống nhau. **Nếu chúng ta cần 7 từ mã sửa lỗi thì đây sẽ là đa thức sinh được yêu cầu.**  
α 0 x 7 + α 87 x 6 + α 229 x 5 + α 146 x 4 + α 149 x 3 + α 238 x 2 + α 102 x 1 + α 21 x 0

*Bước nhân số 7*

Sử dụng đa thức thu được ở bước trước, chúng ta sẽ nhân (α 0 x 7 + α 87 x 6 + α 229 x 5 + α 146 x 4 + α 149 x 3 + α 238 x 2 + α 102 x 1 + α 21 x 0 ) bởi (α 0 x 1 + α 7 x 0 )

Sau khi nhân mỗi số hạng ở phần đầu tiên với mỗi số hạng ở phần thứ hai, chúng ta nhận được:  
(α 0 x 7 × α 0 x 1 ) + (α 87 x 6 × α 0 x 1 ) + (α 229 x 5 × α 0 x 1 ) + (α 146 x 4 × α 0 x 1 ) + (α 149 x 3 × α 0 x 1 ) + (α 238 x 2 × α 0 x 1 ) + (α 102 x 1 × α 0 x 1 ) + (α 21 x 0 × α 0 x 1 ) + (α 0 x 7 × α 7 x 0 ) + (α 87 x 6 × α 7 x 0 ) + (α 229 x 5 × α 7 x 0 ) + (α 146 x 4 × α 7 x 0 ) + (α 149 x 3 × α 7 x 0 ) + (α 238 x 2 × α 7 x 0 ) + (α 102 x 1 × α 7 x 0 ) + ( α 21 x 0 × α 7 x 0 )

Để nhân, bạn cộng các số mũ lại với nhau như thế này:  
(α (0+0) x (7+1) ) + (α (87+0) x (6+1) ) + (α (229+0) x ( 5+1) ) + (α (146+0) x (4+1) ) + (α (149+0) x (3+1) ) + (α (238+0) x (2+1) ) + (α (102+0) x (1+1) ) + (α (21+0) x (0+1) ) + (α (0+7) x (7+0) ) + (α (87 +7) x (6+0) ) + (α (229+7) x (5+0) ) + (α (146+7) x (4+0) ) + (α (149+7) x ( 3+0) ) + (α (238+7) x (2+0) ) + (α (102+7) x (1+0) ) + (α (21+7) x (0+0) )

Sau khi cộng số mũ ta được kết quả:  
α 0 x 8 + α 87 x 7 + α 229 x 6 + α 146 x 5 + α 149 x 4 + α 238 x 3 + α 102 x 2 + α 21 x 1 + α 7 x 7 + α 94 x 6 + α 236 x 5 + α 153 x 4 + α 156 x 3 + α 245 x 2 + α 109 x 1 + α 28 x 0

Nếu có bất kỳ số mũ nào lớn hơn 255, hãy sửa chúng bằng cách đưa chúng vào công thức  
(số mũ % 256) + tầng (số mũ / 256). Trong trường hợp này, tất cả số mũ đều nhỏ hơn hoặc bằng 255 nên chúng ta có thể tiếp tục.

Bây giờ bạn phải kết hợp các số hạng có cùng số mũ. Trong trường hợp này, có nhiều hơn một số hạng x 7 , x 6 , x 5 , x 4 , x 3 , x 2 và x 1 .

Bây giờ chúng ta sẽ kết hợp các số hạng x 7 . chúng là:  
α 87 x 7 + α 7 x 7

Chúng ta sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
127x 7 + 128x 7

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(127 ⊕ 128)x 7 = 255x 7 = α 175 x 7

Bây giờ chúng ta sẽ kết hợp các số hạng x 6 . chúng là:  
α 229 x 6 + α 94 x 6

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
122x 6 + 113x 6

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(122 ⊕ 113)x 6 = 11x 6 = α 238 x 6

Bây giờ chúng ta sẽ kết hợp các số hạng x 5 . chúng là:  
α 146 x 5 + α 236 x 5

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
154x 5 + 203x 5

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(154 ⊕ 203)x 5 = 81x 5 = α 208 x 5

Bây giờ chúng ta sẽ kết hợp các số hạng x 4 . chúng là:  
α 149 x 4 + α 153 x 4

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
164x 4 + 146x 4

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(164 ⊕ 146)x4 = 54x4 = α 249 x 4

Bây giờ chúng ta sẽ kết hợp các số hạng x 3 . chúng là:  
α 238 x 3 + α 156 x 3

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
11x 3 + 228x 3

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(11 ⊕ 228)x 3 = 239x 3 = α 215 x 3

Bây giờ chúng ta sẽ kết hợp các số hạng x 2 . chúng là:  
α 102 x 2 + α 245 x 2

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
68x 2 + 233x 2

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(68 ⊕ 233)x 2 = 173x 2 = α 252 x 2

Bây giờ chúng ta sẽ kết hợp các số hạng x 1 . chúng là:  
α 21 x 1 + α 109 x 1

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
117x 1 + 189x 1

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(117 ⊕ 189)x 1 = 200x 1 = α 196 x 1

Dưới đây là đa thức cuối cùng của bước này sau khi kết hợp các số hạng giống nhau. **Nếu chúng ta cần 8 từ mã sửa lỗi thì đây sẽ là đa thức sinh được yêu cầu.**  
α 0 x 8 + α 175 x 7 + α 238 x 6 + α 208 x 5 + α 249 x 4 + α 215 x 3 + α 252 x 2 + α 196 x 1 + α 28 x 0

*Bước nhân #8*

Sử dụng đa thức thu được ở bước trước, chúng ta sẽ nhân (α 0 x 8 + α 175 x 7 + α 238 x 6 + α 208 x 5 + α 249 x 4 + α 215 x 3 + α 252 x 2 + α 196 x 1 + α 28 x 0 ) bằng (α 0 x 1 + α 8 x 0 )

Sau khi nhân mỗi số hạng ở phần thứ nhất với mỗi số hạng ở phần thứ hai, chúng ta nhận được:  
(α 0 x 8 × α 0 x 1 ) + (α 175 x 7 × α 0 x 1 ) + (α 238 x 6 × α 0 x 1 ) + (α 208 x 5 × α 0 x 1 ) + (α 249 x 4 × α 0 x 1 ) + (α 215 x 3 × α 0 x 1 ) + (α 252 x 2 × α 0 x 1 ) + (α 196 x 1 × α 0 x 1 ) + (α 28 x 0 × α 0 x 1 ) + (α 0 x 8 × α 8 x 0 ) + (α 175 x 7 × α 8 x 0 ) + (α 238 x 6 × α 8 x 0 ) + (α 208 x 5 × α 8 x 0 ) + (α 249 x 4 × α 8 x 0 ) + (α 215 x 3 × α 8 x 0 ) + ( α 252 x 2 × α 8 x 0 ) + (α 196 x 1 × α 8 x 0 ) + (α 28 x 0 × α 8 x 0 )

Để nhân, bạn cộng các số mũ lại với nhau như sau:  
(α (0+0) x (8+1) ) + (α (175+0) x (7+1) ) + (α (238+0) x ( 6+1) ) + (α (208+0) x (5+1) ) + (α (249+0) x (4+1) ) + (α (215+0) x (3+1) ) + (α (252+0) x (2+1) ) + (α (196+0) x (1+1) ) + (α (28+0) x (0+1) ) + (α (0 +8) x (8+0) ) + (α (175+8) x (7+0) ) + (α (238+8) x (6+0) ) + (α (208+8) x ( 5+0) ) + (α (249+8) x (4+0) ) + (α (215+8) x (3+0) ) + (α (252+8) x (2+0) ) + (α (196+8) x (1+0) ) + (α (28+8) x (0+0) )

Sau khi cộng số mũ ta được kết quả:  
α 0 x 9 + α 175 x 8 + α 238 x 7 + α 208 x 6 + α 249 x 5 + α 215 x 4 + α 252 x 3 + α 196 x 2 + α 28 x 1 + α 8 x 8 + α 183 x 7 + α 246 x 6 + α 216 x 5 + α 257 x 4 + α 223 x 3 + α 260 x 2 + α 204 x 1 + α 36 x 0

Nếu có số mũ nào lớn hơn 255, hãy sửa chúng bằng cách đưa chúng vào công thức  
(số mũ % 256) + tầng(số mũ / 256)  
α 0 x 9 + α 175 x 8 + α 238 x 7 + α 208 x 6 + α 249 x 5 + α 215 x 4 + α 252 x 3 + α 196 x 2 + α 28 x 1 + α 8 x 8 + α 183 x 7 + α 246 x 6 + α 216 x 5 + α 2 x 4 + α 223 x 3 + α 5 x 2 + α 204 x 1 + α 36 x 0

Bây giờ bạn phải kết hợp các số hạng có cùng số mũ. Trong trường hợp này, có nhiều hơn một số hạng x 8 , x 7 , x 6 , x 5 , x 4 , x 3 , x 2 và x 1 .

Bây giờ chúng ta sẽ kết hợp các số hạng x 8 . chúng là:  
α 175 x 8 + α 8 x 8

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
255x 8 + 29x 8

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(255 ⊕ 29)x 8 = 226x 8 = α 95 x 8

Bây giờ chúng ta sẽ kết hợp các số hạng x 7 . chúng là:  
α 238 x 7 + α 183 x 7

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
11x 7 + 196x 7

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(11 ⊕ 196)x 7 = 207x 7 = α 246 x 7

Bây giờ chúng ta sẽ kết hợp các số hạng x 6 . chúng là:  
α 208 x 6 + α 246 x 6

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
81x 6 + 207x 6

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(81 ⊕ 207)x 6 = 158x 6 = α 137 x 6

Bây giờ chúng ta sẽ kết hợp các số hạng x 5 . chúng là:  
α 249 x 5 + α 216 x 5

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
54x 5 + 195x 5

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(54 ⊕ 195)x 5 = 245x 5 = α 231 x 5

Bây giờ chúng ta sẽ kết hợp các số hạng x 4 . chúng là:  
α 215 x 4 + α 2 x 4

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
239x 4 + 4x 4

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(239 ⊕ 4)x4 = 235x4 = α 235 x 4

Bây giờ chúng ta sẽ kết hợp các số hạng x 3 . chúng là:  
α 252 x 3 + α 223 x 3

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
173x 3 + 9x 3

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(173 ⊕ 9)x 3 = 164x 3 = α 149 x 3

Bây giờ chúng ta sẽ kết hợp các số hạng x 2 . chúng là:  
α 196 x 2 + α 5 x 2

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
200x 2 + 32x 2

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(200 ⊕ 32)x 2 = 232x 2 = α 11 x 2

Bây giờ chúng ta sẽ kết hợp các số hạng x 1 . chúng là:  
α 28 x 1 + α 204 x 1

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
24x 1 + 221x 1

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(24 ⊕ 221)x 1 = 197x 1 = α 123 x 1

Dưới đây là đa thức cuối cùng của bước này sau khi kết hợp các số hạng giống nhau. **Nếu chúng ta cần 9 từ mã sửa lỗi thì đây sẽ là đa thức sinh được yêu cầu.**  
α 0 x 9 + α 95 x 8 + α 246 x 7 + α 137 x 6 + α 231 x 5 + α 235 x 4 + α 149 x 3 + α 11 x 2 + α 123 x 1 + α 36 x 0

*Bước nhân #9*

Sử dụng đa thức thu được ở bước trước, chúng ta sẽ nhân (α 0 x 9 + α 95 x 8 + α 246 x 7 + α 137 x 6 + α 231 x 5 + α 235 x 4 + α 149 x 3 + α 11 x 2 + α 123 x 1 + α 36 x 0 ) bằng (α 0 x 1 + α 9 x 0 )

Sau khi nhân mỗi số hạng ở phần đầu tiên với mỗi số hạng ở phần thứ hai, chúng ta nhận được:  
(α 0 x 9 × α 0 x 1 ) + (α 95 x 8 × α 0 x 1 ) + (α 246 x 7 × α 0 x 1 ) + (α 137 x 6 × α 0 x 1 ) + (α 231 x 5 × α 0 x 1 ) + (α 235 x 4 × α 0 x 1 ) + (α 149 x 3 × α 0 x 1 ) + (α 11 x 2 × α 0 x 1 ) + (α 123 x 1 × α 0 x 1 ) + (α 36 x 0 × α 0 x 1 ) + (α 0 x 9 × α 9 x 0 ) + (α 95 x 8 × α 9 x 0 ) + (α 246 x 7 × α 9 x 0 ) + (α 137 x 6 × α 9 x 0 ) + (α 231 x 5 × α 9 x 0 ) + ( α 235 x 4 × α 9 x 0 ) + (α 149 x 3 × α 9 x 0 ) + (α 11 x 2 × α 9 x 0 ) + (α 123 x 1 × α 9 x 0 ) + (α 36 x 0 × α 9 x 0 )

Để nhân, bạn cộng các số mũ lại với nhau như sau:  
(α (0+0) x (9+1) ) + (α (95+0) x (8+1) ) + (α (246+0) x ( 7+1) ) + (α (137+0) x (6+1) ) + (α (231+0) x (5+1) ) + (α (235+0) x (4+1) ) + (α (149+0) x (3+1) ) + (α (11+0) x (2+1) ) + (α (123+0) x (1+1) ) + (α (36 +0) x (0+1) ) + (α (0+9) x (9+0) ) + (α (95+9) x (8+0) ) + (α (246+9) x ( 7+0) ) + (α (137+9) x (6+0) ) + (α (231+9) x (5+0) ) + (α (235+9) x (4+0) ) + (α (149+9) x (3+0) ) + (α (11+9) x (2+0) ) + (α (123+9) x (1+0) ) + (α (36 +9) x (0+0) )

Sau khi cộng số mũ ta được kết quả:  
α 0 x 10 + α 95 x 9 + α 246 x 8 + α 137 x 7 + α 231 x 6 + α 235 x 5 + α 149 x 4 + α 11 x 3 + α 123 x 2 + α 36 x 1 + α 9 x 9 + α 104 x 8 + α 255 x 7 + α 146 x 6 + α 240 x 5 + α 244 x 4 + α 158 x 3 + α 20 x 2 + α 132 x 1 + α 45 x 0

Nếu có bất kỳ số mũ nào lớn hơn 255, hãy sửa chúng bằng cách đưa chúng vào công thức  
(số mũ % 256) + tầng (số mũ / 256). Trong trường hợp này, tất cả số mũ đều nhỏ hơn hoặc bằng 255 nên chúng ta có thể tiếp tục.

Bây giờ bạn phải kết hợp các số hạng có cùng số mũ. Trong trường hợp này, có nhiều hơn một số hạng x 9 , x 8 , x 7 , x 6 , x 5 , x 4 , x 3 , x 2 và x 1 .

Bây giờ chúng ta sẽ kết hợp các số hạng x 9 . chúng là:  
α 95 x 9 + α 9 x 9

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
226x 9 + 58x 9

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(226 ⊕ 58)x 9 = 216x 9 = α 251 x 9

Bây giờ chúng ta sẽ kết hợp các số hạng x 8 . chúng là:  
α 246 x 8 + α 104 x 8

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
207x 8 + 13x 8

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(207 ⊕ 13)x 8 = 194x 8 = α 67 x 8

Bây giờ chúng ta sẽ kết hợp các số hạng x 7 . chúng là:  
α 137 x 7 + α 255 x 7

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
158x 7 + 1x 7

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(158 ⊕ 1)x 7 = 159x 7 = α 46 x 7

Bây giờ chúng ta sẽ kết hợp các số hạng x 6 . chúng là:  
α 231 x 6 + α 146 x 6

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
245x 6 + 154x 6

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(245 ⊕ 154)x 6 = 111x 6 = α 61 x 6

Bây giờ chúng ta sẽ kết hợp các số hạng x 5 . chúng là:  
α 235 x 5 + α 240 x 5

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
235x 5 + 44x 5

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(235 ⊕ 44)x 5 = 199x 5 = α 118 x 5

Bây giờ chúng ta sẽ kết hợp các số hạng x 4 . chúng là:  
α 149 x 4 + α 244 x 4

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
164x 4 + 250x 4

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(164 ⊕ 250)x4 = 94x4 = α 70 x 4

Bây giờ chúng ta sẽ kết hợp các số hạng x 3 . chúng là:  
α 11 x 3 + α 158 x 3

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
232x 3 + 183x 3

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(232 ⊕ 183)x 3 = 95x 3 = α 64 x 3

Bây giờ chúng ta sẽ kết hợp các số hạng x 2 . chúng là:  
α 123 x 2 + α 20 x 2

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
197x 2 + 180x 2

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(197 ⊕ 180)x 2 = 113x 2 = α 94 x 2

Bây giờ chúng ta sẽ kết hợp các số hạng x 1 . chúng là:  
α 36 x 1 + α 132 x 1

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
37x 1 + 184x 1

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(37 ⊕ 184)x 1 = 157x 1 = α 32 x 1

Dưới đây là đa thức cuối cùng của bước này sau khi kết hợp các số hạng giống nhau. **Nếu chúng ta cần 10 từ mã sửa lỗi thì đây sẽ là đa thức sinh được yêu cầu.**  
α 0 x 10 + α 251 x 9 + α 67 x 8 + α 46 x 7 + α 61 x 6 + α 118 x 5 + α 70 x 4 + α 64 x 3 + α 94 x 2 + α 32 x 1 + α 45 x 0

*Bước nhân #10*

Sử dụng đa thức thu được ở bước trước, chúng ta sẽ nhân (α 0 x 10 + α 251 x 9 + α 67 x 8 + α 46 x 7 + α 61 x 6 + α 118 x 5 + α 70 x 4 + α 64 x 3 + α 94 x 2 + α 32 x 1 + α 45 x 0 ) bằng (α 0 x 1 + α 10 x 0 )

Sau khi nhân mỗi số hạng ở phần thứ nhất với mỗi số hạng ở phần thứ hai, chúng ta nhận được:  
(α 0 x 10 × α 0 x 1 ) + (α 251 x 9 × α 0 x 1 ) + (α 67 x 8 × α 0 x 1 ) + (α 46 x 7 × α 0 x 1 ) + (α 61 x 6 × α 0 x 1 ) + (α 118 x 5 × α 0 x 1 ) + (α 70 x 4 × α 0 x 1 ) + (α 64 x 3 × α 0 x 1 ) + (α 94 x 2 × α 0 x 1 ) + (α 32 x 1 × α 0 x 1 ) + (α 45 x 0 × α 0 x 1 ) + (α 0 x 10 × α 10 x 0 ) + (α 251 x 9 × α 10 x 0 ) + (α 67 x 8 × α 10 x 0 ) + (α 46 x 7 × α 10 x 0 ) + ( α 61 x 6 × α 10 x 0 ) + (α 118 x 5 × α 10 x 0 ) + (α 70 x 4 × α 10 x 0 ) + (α 64 x 3 × α 10 x 0 ) + (α 94 x 2 × α 10 x 0 ) + (α 32 x 1 × α 10 x 0 ) + (α 45 x 0 × α 10 x 0 )

Để nhân, bạn cộng các số mũ lại với nhau như thế này:  
(α (0+0) x (10+1) ) + (α (251+0) x (9+1) ) + (α (67+0) x ( 8+1) ) + (α (46+0) x (7+1) ) + (α (61+0) x (6+1) ) + (α (118+0) x (5+1) ) + (α (70+0) x (4+1) ) + (α (64+0) x (3+1) ) + (α (94+0) x (2+1) ) + (α (32 +0) x (1+1) ) + (α (45+0) x (0+1) ) + (α (0+10) x (10+0) ) + (α (251+10) x ( 9+0) ) + (α (67+10) x (8+0) ) + (α (46+10) x (7+0) ) + (α (61+10) x (6+0) ) + (α (118+10) x (5+0) ) + (α (70+10) x (4+0) ) + (α (64+10) x (3+0) ) + (α (94 +10) x (2+0) ) + (α (32+10) x (1+0) ) + (α (45+10) x (0+0) )

Sau khi cộng số mũ ta được kết quả:  
α 0 x 11 + α 251 x 10 + α 67 x 9 + α 46 x 8 + α 61 x 7 + α 118 x 6 + α 70 x 5 + α 64 x 4 + α 94 x 3 + α 32 x 2 + α 45 x 1 + α 10 x 10 + α 261 x 9 + α 77 x 8 + α 56 x 7 + α 71 x 6 + α 128 x 5 + α 80 x 4 + α 74 x 3 + α 104 x 2 + α 42 x 1 + α 55 x 0

Nếu có số mũ nào lớn hơn 255, hãy sửa chúng bằng cách đưa chúng vào công thức  
(số mũ % 256) + tầng(số mũ / 256)  
α 0 x 11 + α 251 x 10 + α 67 x 9 + α 46 x 8 + α 61 x 7 + α 118 x 6 + α 70 x 5 + α 64 x 4 + α 94 x 3 + α 32 x 2 + α 45 x 1 + α 10 x 10 + α 6 x 9 + α 77 x 8 + α 56 x 7 + α 71 x 6 + α 128 x 5 + α 80 x 4 + α 74 x 3 + α 104 x 2 + α 42 x 1 + α 55 x 0

Bây giờ bạn phải kết hợp các số hạng có cùng số mũ. Trong trường hợp này, có nhiều hơn một số hạng x 10 , x 9 , x 8 , x 7 , x 6 , x 5 , x 4 , x 3 , x 2 và x 1 .

Bây giờ chúng ta sẽ kết hợp các số hạng x 10 . chúng là:  
α 251 x 10 + α 10 x 10

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
216x 10 + 116x 10

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(216 ⊕ 116)x10 = 172x10 = α 220 x 10

Bây giờ chúng ta sẽ kết hợp các số hạng x 9 . chúng là:  
α 67 x 9 + α 6 x 9

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
194x 9 + 64x 9

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(194 ⊕ 64)x 9 = 130x 9 = α 192 x 9

Bây giờ chúng ta sẽ kết hợp các số hạng x 8 . chúng là:  
α 46 x 8 + α 77 x 8

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
159x 8 + 60x 8

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(159 ⊕ 60)x 8 = 163x 8 = α 91 x 8

Bây giờ chúng ta sẽ kết hợp các số hạng x 7 . chúng là:  
α 61 x 7 + α 56 x 7

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
111x 7 + 93x 7

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(111 ⊕ 93)x 7 = 50x 7 = α 194 x 7

Bây giờ chúng ta sẽ kết hợp các số hạng x 6 . chúng là:  
α 118 x 6 + α 71 x 6

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
199x 6 + 188x 6

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(199 ⊕ 188)x 6 = 123x 6 = α 172 x 6

Bây giờ chúng ta sẽ kết hợp các số hạng x 5 . chúng là:  
α 70 x 5 + α 128 x 5

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
94x 5 + 133x 5

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(94 ⊕ 133)x 5 = 219x 5 = α 177 x 5

Bây giờ chúng ta sẽ kết hợp các số hạng x 4 . chúng là:  
α 64 x 4 + α 80 x 4

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
95x 4 + 253x 4

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(95 ⊕ 253)x4 = 162x4 = α 209 x 4

Bây giờ chúng ta sẽ kết hợp các số hạng x 3 . chúng là:  
α 94 x 3 + α 74 x 3

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
113x 3 + 137x 3

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(113 ⊕ 137)x 3 = 248x 3 = α 116 x 3

Bây giờ chúng ta sẽ kết hợp các số hạng x 2 . chúng là:  
α 32 x 2 + α 104 x 2

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
157x 2 + 13x 2

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(157 ⊕ 13)x 2 = 144x 2 = α 227 x 2

Bây giờ chúng ta sẽ kết hợp các số hạng x 1 . chúng là:  
α 45 x 1 + α 42 x 1

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
193x 1 + 181x 1

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(193 ⊕ 181)x 1 = 116x 1 = α 10 x 1

Dưới đây là đa thức cuối cùng của bước này sau khi kết hợp các số hạng giống nhau. **Nếu chúng ta cần 11 từ mã sửa lỗi thì đây sẽ là đa thức sinh được yêu cầu.**  
α 0 x 11 + α 220 x 10 + α 192 x 9 + α 91 x 8 + α 194 x 7 + α 172 x 6 + α 177 x 5 + α 209 x 4 + α 116 x 3 + α 227 x 2 + α 10 x 1 + α 55 x 0

*Bước nhân #11*

Sử dụng đa thức thu được ở bước trước, chúng ta sẽ nhân (α 0 x 11 + α 220 x 10 + α 192 x 9 + α 91 x 8 + α 194 x 7 + α 172 x 6 + α 177 x 5 + α 209 x 4 + α 116 x 3 + α 227 x 2 + α 10 x 1 + α 55 x 0 ) bằng (α 0 x 1 + α 11 x 0 )

Sau khi nhân mỗi số hạng ở phần thứ nhất với mỗi số hạng ở phần thứ hai, ta được:  
(α 0 x 11 × α 0 x 1 ) + (α 220 x 10 × α 0 x 1 ) + (α 192 x 9 × α 0 x 1 ) + (α 91 x 8 × α 0 x 1 ) + (α 194 x 7 × α 0 x 1 ) + (α 172 x 6 × α 0 x 1 ) + (α 177 x 5 × α 0 x 1 ) + (α 209 x 4 × α 0 x 1 ) + (α 116 x 3 × α 0 x 1 ) + (α 227 x 2 × α 0 x 1 ) + (α 10 x 1 × α 0 x 1 ) + (α 55 x 0 × α 0 x 1 ) + (α 0 x 11 × α 11 x 0 ) + (α 220 x 10 × α 11 x 0 ) + (α 192 x 9 × α 11 x 0 ) + ( α 91 x 8 × α 11 x 0 ) + (α 194 x 7 × α 11 x 0 ) + (α 172 x 6 × α 11 x 0 ) + (α 177 x 5 × α 11 x 0 ) + (α 209 x 4 × α 11 x 0 ) + (α 116 x 3 × α 11 x 0 ) + (α 227 x 2 × α 11 x 0 ) + (α 10 x 1 × α 11 x 0 ) + (α 55 x 0 × α11 x 0 )

Để nhân, bạn cộng các số mũ lại với nhau như sau:  
(α (0+0) x (11+1) ) + (α (220+0) x (10+1) ) + (α (192+0) x ( 9+1) ) + (α (91+0) x (8+1) ) + (α (194+0) x (7+1) ) + (α (172+0) x (6+1) ) + (α (177+0) x (5+1) ) + (α (209+0) x (4+1) ) + (α (116+0) x (3+1) ) + (α (227 +0) x (2+1) ) + (α (10+0) x (1+1) ) + (α (55+0) x (0+1) ) + (α (0+11) x ( 11+0) ) + (α (220+11) x (10+0) ) + (α (192+11) x (9+0) ) + (α (91+11) x (8+0) ) + (α (194+11) x (7+0) ) + (α (172+11) x (6+0) ) + (α (177+11) x (5+0) ) + (α (209 +11) x (4+0) ) + (α (116+11) x (3+0) ) + (α (227+11) x (2+0) ) + (α (10+11) x ( 1+0) ) + (α (55+11) x (0+0) )

Sau khi cộng số mũ ta được kết quả:  
α 0 x 12 + α 220 x 11 + α 192 x 10 + α 91 x 9 + α 194 x 8 + α 172 x 7 + α 177 x 6 + α 209 x 5 + α 116 x 4 + α 227 x 3 + α 10 x 2 + α 55 x 1 + α 11 x 11 + α 231 x 10 + α 203 x 9 + α 102 x 8 + α 205 x 7 + α 183 x 6 + α 188 x 5 + α 220 x 4 + α 127 x 3 + α 238 x 2 + α 21 x 1 + α 66 x 0

Nếu có bất kỳ số mũ nào lớn hơn 255, hãy sửa chúng bằng cách đưa chúng vào công thức  
(số mũ % 256) + tầng (số mũ / 256). Trong trường hợp này, tất cả số mũ đều nhỏ hơn hoặc bằng 255 nên chúng ta có thể tiếp tục.

Bây giờ bạn phải kết hợp các số hạng có cùng số mũ. Trong trường hợp này, có nhiều hơn một số hạng x 11 , x 10 , x 9 , x 8 , x 7 , x 6 , x 5 , x 4 , x 3 , x 2 và x 1 .

Bây giờ chúng ta sẽ kết hợp các số hạng x 11 . chúng là:  
α 220 x 11 + α 11 x 11

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
172x 11 + 232x 11

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(172 ⊕ 232) x11 = 68x11 = α 102 x 11

Bây giờ chúng ta sẽ kết hợp các số hạng x 10 . chúng là:  
α 192 x 10 + α 231 x 10

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
130x 10 + 245x 10

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(130 ⊕ 245) x10 = 119x10 = α 43 x 10

Bây giờ chúng ta sẽ kết hợp các số hạng x 9 . chúng là:  
α 91 x 9 + α 203 x 9

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
163x 9 + 224x 9

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(163 ⊕ 224)x 9 = 67x 9 = α 98 x 9

Bây giờ chúng ta sẽ kết hợp các số hạng x 8 . chúng là:  
α 194 x 8 + α 102 x 8

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
50x 8 + 68x 8

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(50 ⊕ 68)x 8 = 118x 8 = α 121 x 8

Bây giờ chúng ta sẽ kết hợp các số hạng x 7 . chúng là:  
α 172 x 7 + α 205 x 7

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
123x 7 + 167x 7

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(123 ⊕ 167)x 7 = 220x 7 = α 187 x 7

Bây giờ chúng ta sẽ kết hợp các số hạng x 6 . chúng là:  
α 177 x 6 + α 183 x 6

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
219x 6 + 196x 6

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(219 ⊕ 196)x 6 = 31x 6 = α 113 x 6

Bây giờ chúng ta sẽ kết hợp các số hạng x 5 . chúng là:  
α 209 x 5 + α 188 x 5

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
162x 5 + 165x 5

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(162 ⊕ 165)x 5 = 7x 5 = α 198 x 5

Bây giờ chúng ta sẽ kết hợp các số hạng x 4 . chúng là:  
α 116 x 4 + α 220 x 4

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
248x 4 + 172x 4

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(248 ⊕ 172)x4 = 84x4 = α 143 x 4

Bây giờ chúng ta sẽ kết hợp các số hạng x 3 . chúng là:  
α 227 x 3 + α 127 x 3

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
144x 3 + 204x 3

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(144 ⊕ 204)x3 = 92x3 = α 131 x 3

Bây giờ chúng ta sẽ kết hợp các số hạng x 2 . chúng là:  
α 10 x 2 + α 238 x 2

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
116x 2 + 11x 2

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(116 ⊕ 11)x 2 = 127x 2 = α 87 x 2

Bây giờ chúng ta sẽ kết hợp các số hạng x 1 . chúng là:  
α 55 x 1 + α 21 x 1

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
160x 1 + 117x 1

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(160 ⊕ 117)x 1 = 213x 1 = α 157 x 1

Dưới đây là đa thức cuối cùng của bước này sau khi kết hợp các số hạng giống nhau. **Nếu chúng ta cần 12 từ mã sửa lỗi thì đây sẽ là đa thức sinh được yêu cầu.**  
α 0 x 12 + α 102 x 11 + α 43 x 10 + α 98 x 9 + α 121 x 8 + α 187 x 7 + α 113 x 6 + α 198 x 5 + α 143 x 4 + α 131 x 3 + α 87 x 2 + α 157 x 1 + α 66 x 0

*Bước nhân #12*

Sử dụng đa thức thu được ở bước trước, chúng ta sẽ nhân (α 0 x 12 + α 102 x 11 + α 43 x 10 + α 98 x 9 + α 121 x 8 + α 187 x 7 + α 113 x 6 + α 198 x 5 + α 143 x 4 + α 131 x 3 + α 87 x 2 + α 157 x 1 + α 66 x 0 ) bằng (α 0 x 1 + α 12 x 0 )

Sau khi nhân mỗi số hạng ở phần thứ nhất với mỗi số hạng ở phần thứ hai, chúng ta nhận được:  
(α 0 x 12 × α 0 x 1 ) + (α 102 x 11 × α 0 x 1 ) + (α 43 x 10 × α 0 x 1 ) + (α 98 x 9 × α 0 x 1 ) + (α 121 x 8 × α 0 x 1 ) + (α 187 x 7 × α 0 x 1 ) + (α 113 x 6 × α 0 x 1 ) + (α 198 x 5 × α 0 x 1 ) + (α 143 x 4 × α 0 x 1 ) + (α 131 x 3 × α 0 x 1 ) + (α 87 x 2 × α 0 x 1 ) + (α 157 x 1 × α 0 x 1 ) + (α 66 x 0 × α 0 x 1 ) + (α 0 x 12 × α 12 x 0 ) + (α 102 x 11 × α 12 x 0 ) + ( α 43 x 10 × α 12 x 0 ) + (α 98 x 9 × α 12 x 0 ) + (α 121 x 8 × α 12 x 0 ) + (α 187 x 7 × α 12 x 0 ) + (α 113 x 6 × α 12 x 0 ) + (α 198 x 5 × α 12 x 0 ) + (α 143 x 4 × α 12 x 0 ) + (α 131 x 3 × α 12 x 0 ) + (α 87 x 2 × α12 x 0 ) + (α 157 x 1 × α 12 x 0 ) + (α 66 x 0 × α 12 x 0 )

Để nhân, bạn cộng các số mũ lại với nhau như thế này:  
(α (0+0) x (12+1) ) + (α (102+0) x (11+1) ) + (α (43+0) x ( 10+1) ) + (α (98+0) x (9+1) ) + (α (121+0) x (8+1) ) + (α (187+0) x (7+1) ) + (α (113+0) x (6+1) ) + (α (198+0) x (5+1) ) + (α (143+0) x (4+1) ) + (α (131 +0) x (3+1) ) + (α (87+0) x (2+1) ) + (α (157+0) x (1+1) ) + (α (66+0) x ( 0+1) ) + (α (0+12) x (12+0) ) + (α (102+12) x (11+0) ) + (α (43+12) x (10+0) ) + (α (98+12) x (9+0) ) + (α (121+12) x (8+0) ) + (α (187+12) x (7+0) ) + (α (113 +12) x (6+0) ) + (α (198+12) x (5+0) ) + (α (143+12) x (4+0) ) + (α (131+12) x ( 3+0) ) + (α (87+12) x (2+0) ) + (α (157+12) x (1+0) ) + (α (66+12) x (0+0) )

Sau khi cộng số mũ ta được kết quả:  
α 0 x 13 + α 102 x 12 + α 43 x 11 + α 98 x 10 + α 121 x 9 + α 187 x 8 + α 113 x 7 + α 198 x 6 + α 143 x 5 + α 131 x 4 + α 87 x 3 + α 157 x 2 + α 66 x 1 + α 12 x 12 + α 114 x 11 + α 55 x 10 + α 110 x 9 + α 133 x 8 + α 199 x 7 + α 125 x 6 + α 210 x 5 + α 155 x 4 + α 143 x 3 + α 99 x 2 + α 169 x 1 + α 78 x 0

Nếu có bất kỳ số mũ nào lớn hơn 255, hãy sửa chúng bằng cách đưa chúng vào công thức  
(số mũ % 256) + tầng (số mũ / 256). Trong trường hợp này, tất cả số mũ đều nhỏ hơn hoặc bằng 255 nên chúng ta có thể tiếp tục.

Bây giờ bạn phải kết hợp các số hạng có cùng số mũ. Trong trường hợp này, có nhiều hơn một số hạng x 12 , x 11 , x 10 , x 9 , x 8 , x 7 , x 6 , x 5 , x 4 , x 3 , x 2 và x 1 .

Bây giờ chúng ta sẽ kết hợp các số hạng x 12 . chúng là:  
α 102 x 12 + α 12 x 12

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
68x 12 + 205x 12

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(68 ⊕ 205)x 12 = 137x 12 = α 74 x 12

Bây giờ chúng ta sẽ kết hợp các số hạng x 11 . chúng là:  
α 43 x 11 + α 114 x 11

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
119x 11 + 62x 11

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(119 ⊕ 62)x11 = 73x11 = α 152 x 11

Bây giờ chúng ta sẽ kết hợp các số hạng x 10 . chúng là:  
α 98 x 10 + α 55 x 10

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
67x 10 + 160x 10

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(67 ⊕ 160)x 10 = 227x 10 = α 176 x 10

Bây giờ chúng ta sẽ kết hợp các số hạng x 9 . chúng là:  
α 121 x 9 + α 110 x 9

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
118x 9 + 103x 9

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(118 ⊕ 103)x 9 = 17x 9 = α 100 x 9

Bây giờ chúng ta sẽ kết hợp các số hạng x 8 . chúng là:  
α 187 x 8 + α 133 x 8

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
220x 8 + 109x 8

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(220 ⊕ 109)x 8 = 177x 8 = α 86 x 8

Bây giờ chúng ta sẽ kết hợp các số hạng x 7 . chúng là:  
α 113 x 7 + α 199 x 7

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
31x 7 + 14x 7

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(31 ⊕ 14)x 7 = 17x 7 = α 100 x 7

Bây giờ chúng ta sẽ kết hợp các số hạng x 6 . chúng là:  
α 198 x 6 + α 125 x 6

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
7x 6 + 51x 6

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(7 ⊕ 51)x 6 = 52x 6 = α 106 x 6

Bây giờ chúng ta sẽ kết hợp các số hạng x 5 . chúng là:  
α 143 x 5 + α 210 x 5

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
84x 5 + 89x 5

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(84 ⊕ 89)x 5 = 13x 5 = α 104 x 5

Bây giờ chúng ta sẽ kết hợp các số hạng x 4 . chúng là:  
α 131 x 4 + α 155 x 4

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
92x 4 + 114x 4

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(92 ⊕ 114)x 4 = 46x 4 = α 130 x 4

Bây giờ chúng ta sẽ kết hợp các số hạng x 3 . chúng là:  
α 87 x 3 + α 143 x 3

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
127x 3 + 84x 3

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(127 ⊕ 84)x 3 = 43x 3 = α 218 x 3

Bây giờ chúng ta sẽ kết hợp các số hạng x 2 . chúng là:  
α 157 x 2 + α 99 x 2

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
213x 2 + 134x 2

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(213 ⊕ 134)x 2 = 83x 2 = α 206 x 2

Bây giờ chúng ta sẽ kết hợp các số hạng x 1 . chúng là:  
α 66 x 1 + α 169 x 1

Chúng tôi sẽ chuyển đổi các chữ cái ở trên thành số nguyên bằng cách sử dụng [bảng log antilog](https://www.thonky.com/qr-code-tutorial/log-antilog-table) để thực hiện bước này dễ dàng hơn:  
97x 1 + 229x 1

Sau khi XOR các số nguyên, chúng tôi chuyển đổi kết quả trở lại số hạng alpha.  
(97 ⊕ 229)x 1 = 132x 1 = α 140 x 1

Dưới đây là đa thức cuối cùng của bước này sau khi kết hợp các số hạng giống nhau. **Nếu chúng ta cần 13 từ mã sửa lỗi thì đây sẽ là đa thức sinh được yêu cầu.**  
α 0 x 13 + α 74 x 12 + α 152 x 11 + α 176 x 10 + α 100 x 9 + α 86 x 8 + α 100 x 7 + α 106 x 6 + α 104 x 5 + α 130 x 4 + α 218 x 3 + α 206 x 2 + α 140 x 1 + α 78 x 0

Kết quả cuối cùng:  
(α 0 x 13 + α 74 x 12 + α 152 x 11 + α 176 x 10 + α 100 x 9 + α 86 x 8 + α 100 x 7 + α 106 x 6 + α 104 x 5 + α 130 x 4 + α 218 x 3 + α 206 x 2 + α 140 x 1 + α 78 x 0 )

**Giải thích về các mẫu mặt nạ QR**

Khi mã hóa mã QR, có tám mẫu mặt nạ mà bạn có thể sử dụng để thay đổi ma trận xuất ra. Mỗi mẫu mặt nạ thay đổi các bit theo tọa độ của chúng trong ma trận QR. Mục đích của mẫu mặt nạ là giúp máy quét QR dễ đọc mã QR hơn.

*Các mẫu mặt nạ*

Mỗi mẫu mặt nạ sử dụng một công thức để xác định xem có thay đổi màu của bit hiện tại hay không. Bạn đặt tọa độ của bit hiện tại vào công thức và nếu kết quả là 0 thì bạn sử dụng bit đối diện ở tọa độ đó. Ví dụ: nếu bit cho tọa độ (0,3) là 1 và công thức bằng 0 cho tọa độ đó thì bạn đặt 0 tại (0,3) thay vì 1.

Dưới đây là danh sách các công thức mẫu mặt nạ. Lưu ý một số phiên bản chuẩn mã QR có lỗi ở phần mẫu mặt nạ. Thông tin dưới đây đã được sửa chữa.

|  |  |
| --- | --- |
| **Mask Number** | **If the formula below is true for a given row/column coordinate, switch the bit at that coordinate** |
| 0 | (row + column) mod 2 == 0 |
| 1 | (row) mod 2 == 0 |
| 2 | (column) mod 3 == 0 |
| 3 | (row + column) mod 3 == 0 |
| 4 | ( floor(row / 2) + floor(column / 3) ) mod 2 == 0 |
| 5 | ((row × column) mod 2) + ((row × column) mod 3) == 0 |
| 6 | ( ((row × column) mod 2) + ((row × column) mod 3) ) mod 2 == 0 |
| 7 | ( ((row + column) mod 2) + ((row × column) mod 3) ) mod 2 == 0 |

**Bảng chuỗi định dạng và phiên bản**

Mã QR sử dụng mã hóa sửa lỗi và các mẫu mặt nạ. Kích thước của mã QR được biểu thị bằng một con số, gọi là số phiên bản. Để đảm bảo rằng máy quét mã QR giải mã chính xác những gì nó quét, đặc tả mã QR yêu cầu mỗi mã bao gồm một chuỗi thông tin định dạng, chuỗi này cho máy quét mã QR biết mức sửa lỗi và mẫu mặt nạ mà mã QR đang sử dụng. Ngoài ra, đối với phiên bản 7 trở lên, đặc tả mã QR yêu cầu mỗi mã phải bao gồm một chuỗi thông tin phiên bản để cho máy quét mã QR biết mã đó là phiên bản nào. Trang này liệt kê tất cả các chuỗi định dạng và phiên bản có thể có.

*Giới thiệu về chuỗi thông tin định dạng*

Mã QR sử dụng mã hóa sửa lỗi. Đây là cách tạo dữ liệu dư thừa mà máy quét mã QR có thể sử dụng để phát hiện và sửa lỗi trong mã được quét.

Mã QR cũng sử dụng các mẫu mặt nạ. Mẫu mặt nạ là một thuật toán để thay đổi màu sắc (đậm sang sáng hoặc sáng sang tối) của một mẫu pixel nhất định trong mã để giúp máy quét dễ đọc chính xác hơn.

Mã QR phải bao gồm chuỗi định dạng chứa thông tin về mức mã hóa sửa lỗi và mẫu mặt nạ nào đang được sử dụng trong mã. Trang này liệt kê tất cả 32 chuỗi định dạng có thể có. Để có giải thích chi tiết về cách các chuỗi này được tạo, vui lòng truy cập trang [Thông tin định dạng và phiên bản](https://www.thonky.com/qr-code-tutorial/format-version-information) .

*Danh sách tất cả các chuỗi thông tin định dạng*

|  |  |  |
| --- | --- | --- |
| **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 |
| Q | 2 | 011111100110001 |
| Q | 3 | 011101000000110 |
| Q | 4 | 010010010110100 |
| Q | 5 | 010000110000011 |
| Q | 6 | 010111011011010 |
| Q | 7 | 010101111101101 |
| H | 0 | 001011010001001 |
| H | 1 | 001001110111110 |
| H | 2 | 001110011100111 |
| H | 3 | 001100111010000 |
| H | 4 | 000011101100010 |
| H | 5 | 000001001010101 |
| H | 6 | 000110100001100 |
| H | 7 | 000100000111011 |

*Giới thiệu về chuỗi thông tin phiên bản*

Kích thước của mã QR được biểu thị bằng một số gọi là số phiên bản. Mã phiên bản 7 trở lên phải bao gồm hai khối hình chữ nhật 6x3 chứa chuỗi thông tin phiên bản. Để biết chi tiết về cách tính các chuỗi thông tin phiên bản này và vị trí đặt chúng trong mã QR, vui lòng xem trang [Thông tin định dạng và phiên bản](https://www.thonky.com/qr-code-tutorial/format-version-information) .

|  |  |
| --- | --- |
| **Version** | **Version Information String** |
| 7 | 000111110010010100 |
| 8 | 001000010110111100 |
| 9 | 001001101010011001 |
| 10 | 001010010011010011 |
| 11 | 001011101111110110 |
| 12 | 001100011101100010 |
| 13 | 001101100001000111 |
| 14 | 001110011000001101 |
| 15 | 001111100100101000 |
| 16 | 010000101101111000 |
| 17 | 010001010001011101 |
| 18 | 010010101000010111 |
| 19 | 010011010100110010 |
| 20 | 010100100110100110 |
| 21 | 010101011010000011 |
| 22 | 010110100011001001 |
| 23 | 010111011111101100 |
| 24 | 011000111011000100 |
| 25 | 011001000111100001 |
| 26 | 011010111110101011 |
| 27 | 011011000010001110 |
| 28 | 011100110000011010 |
| 29 | 011101001100111111 |
| 30 | 011110110101110101 |
| 31 | 011111001001010000 |
| 32 | 100000100111010101 |
| 33 | 100001011011110000 |
| 34 | 100010100010111010 |
| 35 | 100011011110011111 |
| 36 | 100100101100001011 |
| 37 | 100101010000101110 |
| 38 | 100110101001100100 |
| 39 | 100111010101000001 |
| 40 | 101000110001101001 |

**Bảng giá trị chữ và số**

Khi tạo mã QR ở chế độ chữ và số, bạn cần sử dụng giá trị chữ và số của các ký tự trong quá trình mã hóa. Bảng này cung cấp tham chiếu cho các ký tự được hỗ trợ ở chế độ chữ và số cho mã QR.

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

**Mã hóa chế độ Kanji**

*Chỉ dành cho các ký tự JIS Shift đôi byte*

Chế độ Kanji chỉ có thể mã hóa các ký tự Shift JIS hai byte có byte nằm trong phạm vi 0x8140 đến 0x9FFC và 0xE040 đến 0xEBBF (thập lục phân). Các ký tự trong bộ này có thể được tìm thấy trên [Bảng mã Kanji Shift JIS](http://www.rikai.com/library/kanjitables/kanji_codes.sjis.shtml) của Rikai .

*Chuyển đổi sang byte*

Để mã hóa các ký tự Shift JIS 2 byte ở chế độ Kanji cho mã QR, trước tiên các ký tự này phải được chuyển đổi thành byte. Ví dụ: ký tự 茗荷 có thể được chuyển đổi như sau.

茗 → 0xE4AA  
荷 → 0x89D7

*Mã hóa byte bằng chế độ Kanji*

Chế độ Kanji có hai phương pháp mã hóa các ký tự Kanji hai byte. Một phương pháp dành cho các ký tự có byte nằm trong phạm vi 0x8140 đến 0x9FFC. Phương pháp khác dành cho các ký tự có byte nằm trong phạm vi 0xE040 đến 0xEBBF. Hai phương pháp này được trình bày dưới đây.

Đối với các ký tự có byte nằm trong phạm vi 0x8140 đến 0x9FFC

Từ ví dụ trên, ký tự 荷 là 0x89D7 trong Shift JIS, vì vậy nó nằm trong phạm vi 0x8140 đến 0x9FFC. Bước đầu tiên là trừ 0x8140 khỏi giá trị hex. Trong ví dụ này:

0x89D7 - 0x8140 = **0x0897**

Kết quả là 0x0897. Chia số đó thành byte có ý nghĩa nhất và byte có ý nghĩa nhỏ nhất:

Byte quan trọng nhất của 0x0897 là 0x08  
Byte quan trọng nhất của 0x0897 là 0x97

Tiếp theo, nhân byte có ý nghĩa lớn nhất với 0xC0, sau đó thêm byte có ý nghĩa nhỏ nhất vào kết quả:

(0x08 × 0xC0) + 0x97 = (0x600) + 0x97 = 0x697

Cuối cùng, chuyển đổi kết quả đó thành nhị phân 13 bit:

0x697 = 0 0110 1001 0111

Đối với các ký tự có byte nằm trong phạm vi 0xE040 đến 0xEBBF

Từ ví dụ trên, ký tự 茗 là 0xE4AA trong Shift JIS, vì vậy nó nằm trong phạm vi 0xE040 đến 0xEBBF. Bước đầu tiên là trừ 0xC140 khỏi giá trị hex. Trong ví dụ này:

0xE4AA - 0xC140 = **0x236A**

Kết quả là 0x236A. Chia số đó thành byte có ý nghĩa nhất và byte có ý nghĩa nhỏ nhất:

Byte quan trọng nhất của 0x236A là 0x23  
Byte quan trọng nhất của 0x236A là 0x6A

Tiếp theo, nhân byte có ý nghĩa lớn nhất với 0xC0, sau đó thêm byte có ý nghĩa nhỏ nhất vào kết quả:

(0x23 × 0xC0) + 0x6A = (0x1A40) + 0x6A = 0x1AAA

Cuối cùng, chuyển đổi kết quả đó thành nhị phân 13 bit:

0x1AAA = 1 1010 1010 1010

*Đặt các số nhị phân 13 bit lại với nhau*

Chuỗi được mã hóa là 茗荷. Hai ký tự này đã được chuyển đổi thành nhị phân 13 bit trong các bước trên. Đặt hai chuỗi nhị phân lại với nhau để hoàn thành việc mã hóa.

茗荷 được mã hóa cho mã QR là 11010101010100011010010111

**Mã hóa chế độ chữ và số**

Để mã hóa chế độ chữ và số, tôi sẽ sử dụng đầu vào mẫu của **HELLO WORLD** . Trong ví dụ này, tôi sẽ sử dụng phiên bản 1. **Hãy nhớ rằng, chế độ chữ và số chỉ có thể mã hóa chữ hoa chứ không phải chữ thường.**Tham khảo [bảng chữ và số](https://www.thonky.com/qr-code-tutorial/alphanumeric-table) để biết danh sách các ký tự có thể được mã hóa ở chế độ chữ và số.

*Chia thành từng cặp*

Đầu tiên chia chuỗi thành các cặp ký tự: HE, LL, O, WO, RL, D

*Tạo số nhị phân cho mỗi cặp*

Đối với chế độ chữ và số, mỗi ký tự chữ và số được biểu thị bằng một số. Vui lòng tham khảo [bảng chữ và số](https://www.thonky.com/qr-code-tutorial/alphanumeric-table) để tìm những con số này. Cột bên trái hiển thị ký tự chữ và số và cột bên phải hiển thị số đại diện cho ký tự đó.

Với mỗi cặp ký tự, lấy cách biểu diễn số (từ [bảng chữ và số](https://www.thonky.com/qr-code-tutorial/alphanumeric-table) ) của ký tự đầu tiên rồi nhân với 45. Sau đó cộng số đó vào cách biểu diễn số của ký tự thứ hai.

Ví dụ: cặp đầu tiên trong HELLO WORLD là HE.

H → 17  
E → 14

Làm theo các bước trong đoạn trước, nhân số đầu tiên với 45, sau đó cộng số đó với số thứ hai:

(45 × 17) + 14 = 779

Bây giờ hãy chuyển đổi số đó thành chuỗi nhị phân 11 bit, đệm bên trái bằng số 0 nếu cần.

779 → 01100001011

Nếu bạn đang mã hóa một số ký tự lẻ, như chúng tôi đang ở đây, hãy lấy biểu diễn số của ký tự cuối cùng và chuyển đổi nó thành chuỗi nhị phân 6 bit.

*Tiếp theo: Hoàn tất bước mã hóa dữ liệu*

Làm theo hướng dẫn trên phần [mã hóa dữ liệu](https://www.thonky.com/qr-code-tutorial/data-encoding) để thêm bất kỳ bit nào còn lại nếu cần.

**Mã hóa chế độ số**

Để minh họa mã hóa chế độ số, đầu vào mẫu là **8675309** , được mã hóa bằng mã QR phiên bản 1.

*Bước 1: Chia chuỗi thành nhóm ba*

Để mã hóa một chuỗi chữ số ở chế độ số, trước tiên hãy chia chuỗi thành các nhóm gồm ba chữ số. Nếu độ dài của chuỗi không phải là bội số của 3 thì nhóm chữ số cuối cùng sẽ chỉ dài một hoặc hai số.

**Sau khi chia nhóm 3 người:**  
867 530 9

*Bước 2: Chuyển đổi từng nhóm thành nhị phân*

Bây giờ hãy coi mỗi nhóm chữ số là một số có ba chữ số (hoặc ít hơn ba, nếu nhóm cuối cùng dài hai hoặc một chữ số). Chuyển số có ba chữ số đó thành 10 bit nhị phân. Nếu một nhóm bắt đầu bằng số 0 thì nó phải được hiểu là số có hai chữ số và bạn nên chuyển đổi nó thành 7 bit nhị phân và nếu có hai số 0 ở đầu nhóm thì nó phải được hiểu là một chữ số số và bạn nên chuyển đổi nó thành 4 bit nhị phân. Tương tự, nếu nhóm cuối cùng chỉ gồm hai chữ số thì nên chuyển đổi thành 7 bit nhị phân, còn nếu nhóm cuối cùng chỉ gồm một chữ số thì nên chuyển đổi thành 4 bit nhị phân.

**Chuyển đổi sang nhị phân:**

867 → 1101100011

530 → 1000010010

9 → 1001

*Tiếp theo: Hoàn tất bước mã hóa dữ liệu*

Làm theo hướng dẫn trên phần [mã hóa dữ liệu](https://www.thonky.com/qr-code-tutorial/data-encoding) để thêm bất kỳ bit nào còn lại nếu cần.

**Mã hóa chế độ byte**

Ở chế độ byte, chuỗi dữ liệu bao gồm chỉ báo chế độ, chỉ báo đếm ký tự và sau đó là các byte thô từ văn bản đầu vào.

*Chuyển đổi sang ISO 8859-1 hoặc UTF-8*

Bộ ký tự mặc định cho chế độ byte là ISO 8859-1 và khi có thể, bạn nên chuyển đổi văn bản đầu vào của mình sang bộ ký tự này. Đặc tả mã QR thảo luận về chế độ ECI, cho phép bạn chỉ định một bộ ký tự khác với ISO 8859-1, nhưng một số trình đọc mã QR không hiểu các chuỗi thoát ECI.

Nếu có các ký tự trong chuỗi đầu vào không thể mã hóa theo ISO 8859-1 thì thay vào đó bạn có thể mã hóa nó theo UTF-8 vì một số trình đọc mã QR có thể phát hiện và hiển thị chính xác mã hóa UTF-8 trong chế độ byte mà không yêu cầu bất kỳ chuỗi thoát ECI nào.

Để giải quyết vấn đề này, bạn có thể muốn kiểm tra các trình đọc mã QR khác nhau để tìm hiểu cách chúng xử lý các ký tự không phải ISO 8859-1 ở chế độ byte hoặc yêu cầu người dùng của bạn cung cấp phản hồi về trình đọc mã QR mà họ sử dụng.

*Chia chuỗi thành byte 8 bit*

Sau khi chuyển đổi chuỗi đầu vào của bạn thành ISO 8859-1 hoặc UTF-8 nếu người dùng của bạn có trình đọc mã QR có thể nhận dạng chuỗi đó ở chế độ byte, bạn phải chia chuỗi thành byte 8 bit.

Ví dụ: chúng tôi sẽ sử dụng chuỗi đầu vào "Xin chào thế giới!" để tạo mã QR phiên bản 1. Vì nó chứa các chữ cái viết thường, dấu phẩy và dấu chấm than nên không thể mã hóa nó bằng chế độ chữ và số, chế độ này không bao gồm các chữ cái viết thường, dấu phẩy hoặc dấu chấm than.

**Chuỗi ví dụ được chuyển đổi thành byte thập lục phân:**  
H → 0x48  
e → 0x65  
l → 0x6c  
l → 0x6c  
o → 0x6f  
, → 0x2c  
  → 0x20  
w → 0x77  
o → 0x6f  
r → 0x72  
l → 0x6c  
d → 0x64  
! → 0x21*Chuyển đổi từng byte thành nhị phân*

Chuyển đổi byte thành chuỗi nhị phân 8 bit. Thêm số 0 vào bên trái nếu cần thiết để làm cho mỗi số có độ dài 8 bit.

**Chuỗi ví dụ với mỗi byte được chuyển đổi thành chuỗi nhị phân 8 bit:**  
H → 0x48 → 01001000  
e → 0x65 → 01100101  
l → 0x6c → 01101100  
l → 0x6c → 01101100  
o → 0x6f → 01101111  
, → 0x2c → 00101100  
  → 0x20 → 00100000  
w → 0x77 → 01110111  
o → 0x6f → 01101111  
r → 0x72 → 01110010  
l → 0x6c → 01101100  
d → 0x64 → 01100100  
! → 0x21 → 00100001*Tiếp theo: Hoàn tất bước mã hóa dữ liệu*

Làm theo hướng dẫn trên phần [mã hóa dữ liệu](https://www.thonky.com/qr-code-tutorial/data-encoding) để thêm bất kỳ bit nào còn lại nếu cần.

**Bảng vị trí mẫu căn chỉnh**

Các mẫu căn chỉnh được giải thích chi tiết về [vị trí mô-đun trong](https://www.thonky.com/qr-code-tutorial/module-placement-matrix) trang ma trận. Các số ở bên phải của bảng này sẽ được sử dụng làm tọa độ CẢ hàng và cột. Ví dụ: Phiên bản 2 có các số 6 và 18. Điều này có nghĩa là các mẫu căn chỉnh sẽ được đặt tại (6, 6), (6, 18), (18, 6) và (18, 18). Tuy nhiên, không đặt các mẫu căn chỉnh lên trên các mẫu tìm kiếm hoặc dấu phân cách.

*Vị trí mẫu căn chỉnh*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Version** | **Center Module Row and Column** | | | | | | |
| 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 |  |  |  |
| QR Version 18 | 6 | 30 | 56 | 82 |  |  |  |
| QR Version 19 | 6 | 30 | 58 | 86 |  |  |  |
| QR Version 20 | 6 | 34 | 62 | 90 |  |  |  |
| QR Version 21 | 6 | 28 | 50 | 72 | 94 |  |  |
| QR Version 22 | 6 | 26 | 50 | 74 | 98 |  |  |
| QR Version 23 | 6 | 30 | 54 | 78 | 102 |  |  |
| QR Version 24 | 6 | 28 | 54 | 80 | 106 |  |  |
| QR Version 25 | 6 | 32 | 58 | 84 | 110 |  |  |
| QR Version 26 | 6 | 30 | 58 | 86 | 114 |  |  |
| QR Version 27 | 6 | 34 | 62 | 90 | 118 |  |  |
| QR Version 28 | 6 | 26 | 50 | 74 | 98 | 122 |  |
| QR Version 29 | 6 | 30 | 54 | 78 | 102 | 126 |  |
| QR Version 30 | 6 | 26 | 52 | 78 | 104 | 130 |  |
| QR Version 31 | 6 | 30 | 56 | 82 | 108 | 134 |  |
| QR Version 32 | 6 | 34 | 60 | 86 | 112 | 138 |  |
| QR Version 33 | 6 | 30 | 58 | 86 | 114 | 142 |  |
| QR Version 34 | 6 | 34 | 62 | 90 | 118 | 146 |  |
| QR Version 35 | 6 | 30 | 54 | 78 | 102 | 126 | 150 |
| QR Version 36 | 6 | 24 | 50 | 76 | 102 | 128 | 154 |
| QR Version 37 | 6 | 28 | 54 | 80 | 106 | 132 | 158 |
| QR Version 38 | 6 | 32 | 58 | 84 | 110 | 136 | 162 |
| QR Version 39 | 6 | 26 | 54 | 82 | 110 | 138 | 166 |
| QR Version 40 | 6 | 30 | 58 | 86 | 114 | 142 | 170 |