Cyclic Mixed-Radix Dense Gray Codes

Jessica Cheng
Dartmouth College

Follow this and additional works at: https://digitalcommons.dartmouth.edu/cs_senior_theses

Part of the Theory and Algorithms Commons

Recommended Citation
Cheng, Jessica, "Cyclic Mixed-Radix Dense Gray Codes" (2023). Computer Science Senior Theses. 22.
https://digitalcommons.dartmouth.edu/cs_senior_theses/22

This Thesis (Undergraduate) is brought to you for free and open access by the Computer Science at Dartmouth Digital Commons. It has been accepted for inclusion in Computer Science Senior Theses by an authorized administrator of Dartmouth Digital Commons. For more information, please contact dartmouthdigitalcommons@groups.dartmouth.edu.
Cyclic Mixed-Radix Dense Gray Codes

Jessica Cheng

jessica.cheng.23@dartmouth.edu

Dartmouth College Department of Computer Science
Gray code for $n=4$  
| 00  |
| 01  |
| 11  |
| 10  |
| 110 |
| 111 |
| 101 |
| 100 |

Gray code for $n=8$  
| 000 |
| 001 |
| 011 |
| 010 |
| 110 |
| 111 |
| 101 |
| 100 |

Table 1: The standard binary reflected Gray codes for $n=4$ and $n=8$.

1 Introduction

In 1953, Frank Gray patented the standard binary reflected Gray code [2]. A Gray code is a sequence of $n$ binary integers in the range 0 to $n-1$ that has the Gray-code property: each integer in the sequence differs from the integer before it in a single digit. Gray codes have many applications, ranging from rotary encoders [4] to Boolean circuit minimization [5]. The standard binary reflected Gray code, however, only allows for values of $n$ that are a power of 2. Table 1 shows the Gray codes for $n=4$ and $n=8$. We refer to each number in the Gray code sequence as a codeword. Notice that these Gray codes are represented with 2 and 3 bits respectively, since the number of bits in each sequence is the same as the number of bits needed to represent $n-1$, the largest codeword in the sequence. For both of these Gray codes, the first and last codewords in the sequence also fulfill the Gray-code property. We refer to Gray codes with this characteristic as cyclic. Additionally, we refer to a Gray code as dense if the sequence of $n$ numbers consists of a permutation of $(0,1,\ldots,n-1)$.

Even though the standard binary reflected Gray code only applies to values of $n$ that are powers of 2 and binary numbers, the property can be expanded to other values of $n$ and other radices. For example, suppose we choose to represent numbers in base 5. With this fixed-radix representation of $(5,5,5,5)$, we see that codewords 0314 and 0214 satisfy the Gray-code property because they differ by 1 in a single digit, in this case the second-most-significant digit. Codewords in a Gray code sequence may also be modular cyclic, meaning that adjacent codewords in radix $r$ can satisfy the Gray-code property if they vary in a digit with values 0 and $r-1$. For example, in base 5, codewords 0110 and 0114 satisfy the Gray-code property because 0 and 4 are modular cyclic in base 5. We refer to this modular cyclic property as a modular jump for a specific radix.

The Gray-code property can also be applied to mixed radices. Mixed radix is a base representation that uses any $d$-tuple of radices $r=(r_{d-1},r_{d-2},\ldots,r_0)$. In mixed-radix notation, each digit position can have a different base. Even though mixed radix may seem unfamiliar, we write time in mixed-radix notation where $r=(24,60,60)$. The most significant radix, 24, represents hours, and in each hour there are 60 minutes, each of which has 60 seconds. For any given mixed
radix, we use \( d \) to refer to the total number of radices. For example, \( d = 4 \) for radices \((2, 3, 4, 5)\). Additionally, we assign each digit a \textit{position} from right to left starting with 0. For example, for radices \((2, 3, 4, 5)\), radix 5 is position 0, radix 4 is position 1, radix 3 is position 2, and radix 2 is position 3. We can also refer to radix 2 as digit \( d - 1 \) and so on.

We define \( p_i = \prod_{j=0}^{i} r_j \) and \( p_{-1} = 1 \). The value of a mixed-radix number \((x_{d-1}, x_{d-2}, \ldots, x_0)\) is then \( \sum_{i=0}^{d-1} x_i p_{i-1} \). Using the time example, a time of \( 2 : 15 : 30 \) has a value of \( 2 \cdot 3600 + 15 \cdot 60 + 30 \cdot 1 = 8130 \), or 2 hours, 15 minutes, and 30 seconds is equal to 8130 seconds.

This thesis builds on the previous work of thesis students Jessica Fan and Devina Kumar. Their previous results provide a solution to generate a cyclic, mixed-radix, dense Gray code except when all three of the following hold: \( r_{d-1} = 2 \), there is at least one odd radix, and \( n \) is odd. This thesis will address creating a dense, cyclic Gray code for mixed radices \( r \) when \( r_{d-1} = 2 \), there is at least one odd radix, and \( n \) is odd. Section 2 introduces previous research done on this topic by previous thesis students and the relevant results from their theses. Section 3 of this thesis discusses an overall approach and intuition to creating cyclic, mixed-radix, dense Gray codes that fall into this category. I consider only odd values of \( n \) that do not fall under Fan’s impossibility results in Theorem 1 and 2, that is when \( p_{d-2} + 1 < n < p_{d-1} \). Given these criteria, this thesis provides a method to generate \( G \), a cyclic, mixed-radix, dense Gray code in the following situations and their corresponding sections:

- Section 4: All values of \( n \) when \( r_{d-2}, \ldots, r_0 \) are odd
- Section 5: Values of \( n \) where \( p_{d-2} + r_0 \leq n < p_{d-1} \) when \( r_0 \) is odd and at least one of \( r_{d-2}, \ldots, r_1 \) is even
- Section 6: Values of \( n \) where \( p_{d-2} + r_0 + 1 \leq n < p_{d-1} \) and \( d = 3 \) when \( r_0 \) is even

2 Background

In Fan’s 2017 honors thesis [1], Fan developed a recursive algorithm to generate a cyclic mixed-radix full Gray code for \( d \) radices. She also found and proved two scenarios where generating a cyclic mixed-radix dense Gray code is not possible as shown in the following theorems.

Fan represents the Gray code as a \textit{modular Gray graph} for radix tuple \( r \) and \( n \) codewords. The graph is an undirected graph \( G = (V, E) \) where \( |V| = n \), each vertex \( v \in V \) is a unique codeword from \( \{0, 1, \ldots, n-1\} \), \( E \) is the set of edges \((u, v)\) where \( u, v \in V \), and the codewords at \( u \) and \( v \) satisfy the Gray-code property. Using the modular Gray graph, finding a cyclic, mixed-radix, dense Gray code is the equivalent of finding a Hamiltonian cycle—a cycle that visits each vertex \( v \in V \) exactly once. When \( n = (q, 0, 0, \ldots, 0, 1) \), either vertex 0 or vertex \( n-1 \) has degree 1, making it impossible to produce a Hamiltonian cycle and thus impossible to create a cyclic, dense Gray code as stated in Theorem 1.
Figure 1: A cycle for $r_i$ showing all the possible digit values in position $i$. This is equivalent to the set of integers $0, 1, \ldots, r_i - 1$ in a circle.

**Theorem 1** Graphs with vertices of degree 1 are non-Hamiltonian

Let $n$ be a positive integer, and let $d$ be the number of digits required to represent $n - 1$ using the radix tuple $r = (r_{d-1}, r_{d-2}, \ldots, r_0)$, so that the digit representation of $n - 1$ is $(x_{d-1}, x_{d-2}, \ldots, x_0)$, where $n_{d-1} > 0$. Let $G$ be the modular Gray graph for $r$ and $n$. If $n$ has the digit representation

$$n = q \, 0 \, 0 \, \cdots \, 0 \, 1,$$

(1)

where either $q = 1$ or $q < r_{d-1} - 1$, then $G$ has no Hamiltonian cycle.

**Theorem 2** Impossibility of both cyclicity and density for some values of $r$ and $n$

Let $n$ be a positive integer, and let $d$ be the number of digits required to represent $n - 1$ using the radix tuple $r = (r_{d-1}, r_{d-2}, \ldots, r_0)$, so that the digit representation of $n - 1$ is $(x_{d-1}, x_{d-2}, \ldots, x_0)$, where $n_{d-1} > 0$. Let $G$ be the modular Gray graph for $r$ and $n$. If $n$ is odd and $r_i$ is even for each digit position $i = 0, 1, \ldots, d - 2$, and either $r_{d-1}$ is even or $n_{d-1} < r_{d-1} - 1$, then $G$ has no Hamiltonian cycle.

In other words, Fan states that a cyclic, dense Gray code is impossible when every radix in $r$ is even and $n$ is odd. For a Gray code to be cyclic, it must start and end at the same codeword. For any radix $r_i$ where $i = 0, 1, \ldots, d - 1$, if $r_i$ is even, the number of digit changes made in position $i$ must also be even if we start and end at the same number. Figure 1 shows the cycle for $r_i$, representing all the possibilities a digit in position $i$ can take in a circle. We see that if we want to start and end at the same place in the circle, we must traverse an even amount of edges. Given that every radix in $r$ is even, there is no way to start and end at the same codeword such that an odd number of edges have been traversed. I now introduce a corollary to Fan’s Theorem 2.

**Corollary 3** If $G$ is a cyclic, mixed-radix, dense Gray code with radices $r = (r_{d-1}, r_{d-2}, \ldots, r_0)$ where at least one radix is odd and $n$ is odd, then it must have an odd number of modular jumps in odd radices.
Proof: As illustrated in Fan’s Theorem 2, there must be an even number of digit changes for any even radix \( r_i \) to start and end on the same value. For any odd radix \( r_i \), if we start and end a sequence on the same value without a modular jump, there must also be an even number of digit changes. We can imagine that any steps taken along the ring in Figure 1 that are not a modular jump must necessarily be backtracked and undone to arrive at the starting number. This backtracking results in an even number of digit changes. The only way to have an odd number of digit changes is to use the modular jump in an odd radix to create the cyclic property. Additionally, if we have an even number of modular jumps in an odd radix, the total number of digit changes will be even because an even number of odd numbers sums to an even number. Thus, to have an odd number of codewords, there must be an odd number of modular jumps in odd radices.

The impossibility results from Fan’s thesis inform the values of \( n \) that we consider in this thesis. In Section 3.2 of Kumar’s thesis [3], she introduces the stitching method to create a cyclic, dense Gray code. We will defer an explanation of this method until later on, but I use it in the solution outlined in this paper.

3 Overall Approach

This section provides the approach and intuition that all solutions detailed in the following sections will utilize to create a cyclic, dense Gray code. As shown in Section 2, there must be an odd number of modular jumps in an odd radix. My approach will have just a single modular jump that occurs in the rightmost odd radix. We refer to the position of the rightmost odd radix as position \( m \), where \( r_m \) is the value of the rightmost odd radix.

To create our Gray code, we start with the reflected cyclic Gray code as laid out in Fan’s thesis and make certain modifications to create a base configuration. The base configuration is the configuration of codewords that our cyclic, dense Gray codes for all values of \( n \) will be based off of. The reflected cyclic Gray code is shown for radices \((2, 3, 3)\) in Figure 2(a). To ensure that we have a modular jump at position \( m \), we move all codewords with a value of 0 in position \( m \) to the bottom of the Gray code in the reverse order that they appear in the reflected Gray code. In the example with radices \((2, 3, 3)\), the rightmost odd radix is in position 0, so that we move all codewords ending with 0 to the bottom as shown in Figure 2(b).

Moving the codewords splits our configuration into an upper portion—where all the codewords are non-zero in position \( m \), and a lower portion—where all the codewords are 0 in position \( m \). The modular jump occurs in position \( m \) between the upper and lower portions. We refer to the separation between the upper and lower portions as the dividing line. The dividing line can be seen in purple in Figure 2(c), as well as the codewords where the modular jump occurs, which are highlighted in orange.
Figure 2: The overall approach to creating a cyclic, dense Gray code illustrated with $(2, 3, 3)$. (a) The reflected cyclic Gray code. (b) The base configuration for radices $(2, 3, 3)$ with all codewords ending in 0 moved to the bottom. (c) The base configuration with the dividing line and modular jump highlighted in purple and orange, respectively. (d) The base configuration with a knockout group for $n = 13 = (1, 1, 1)$ in black. (e) The reflected cyclic Gray code from part (a) with all digits in position 0 reflected.

To make a cyclic, dense Gray code for any odd $n$ in a valid range, we knock out any codewords in the reflected cyclic Gray code that are greater than or equal to $n$. These groups of knocked out codewords are made up of one or more consecutive codewords and we refer to these groups as knockout groups. We will refer to any codewords not in the knockout group, which are less than $n$, as remaining codewords. Figure 2(d) shows the knockout group in black for $(2, 3, 3)$ with $n = 13 = (1, 1, 1)$. Depending on the value of $n$, there can be multiple knockout groups in both the upper and lower portions of our configuration. In Figure 2(d) there is one contiguous knockout group that spans the upper and lower portions of the configuration.

To use the stitching method to create the Gray code, we must avoid odd isolated groups of remaining codewords between two knockout groups. We can reflect digits in certain positions to avoid these odd isolated groups. In a radix with value $r$, the reflected value of a digit $q$ is $r - q - 1$. For example, with a radix of 6, every value of 0 becomes 5, 1 becomes 4, 2 becomes 3, and so on. Figure 2(e) shows Figure 2(a) with all digits in position 0 reflected.

The goal of changing our base configuration by reflecting digits is to create contiguous knockout groups in the upper and lower portions. In fact, we want to change our base configuration so that we have a single contiguous knockout group. The knockout group will necessarily span the dividing line if the knockout groups in both the upper and lower portions are not empty. Creating one contiguous knockout group will ensure that we can use the stitching method to create our cyclic, dense Gray code for reasons we will see later.

To create the contiguous knockout group, we identify the largest remaining codeword, which has a value of $n - 1$. If $n - 1$ is in the upper portion, we reflect
columns until it is the remaining codeword right above the knockout group. If \( n - 1 \) is in the lower portion, we reflect columns until it is the remaining codeword right below the knockout group.

4 Solution for Leading Radix of 2 Followed by All Odd Radices

Radices that fall into this category include \((2, 3, 5)\) and \((2, 9, 13, 17, 3)\). Before moving into the steps to create a cyclic, dense Gray code for this section, we must first define a few terms. In our base configuration, each digit in each codeword is part of a sequence. These sequences consist of a series of codewords and have a direction that is either ascending or descending. In Figure 3(a), sequences in position 0 for \((2, 5, 5)\) are highlighted. Ascending sequences are highlighted in red, while descending sequences are highlighted in blue. Below the dividing line, the sequence in position 0 has no direction because all digits there have a value of 0. Figure 3(b) shows the sequences in position 1. We see that in this position, all digits in the upper portion are in an ascending sequence and all digits in the lower portion are in a descending sequence.

I will show the process to make this Gray code using example radices \((2, 5, 5)\) and \((2, 3, 3, 5)\). We can split the process into the following steps as seen below.

1. Make the base configuration
   (a) Create the reflected cyclic Gray code
       Starting with the reflected cyclic Gray code laid out in Fan’s thesis, split this Gray code into two columns: one starting with 0 and one starting with 1, as illustrated for \((2, 5, 5)\) in Figure 4(a).
   (b) Move all codewords ending in 0 to the bottom
       Next, take all the codewords that end in 0 and move them to the bottom of the configuration in the opposite order that they appear in the reflected code, as illustrated in Figure 4(b). This configuration is our base configuration. Note that the modular jump in an odd radix occurs in position 0 between the upper and lower portions, as highlighted in orange.

2. Create knockout groups
   We can then make knockout groups, consisting of all codewords that are greater than or equal to \( n \). Figure 4(c) illustrates the knockout groups for \((2, 5, 5)\) with \( n = 33 = (1, 1, 3) \). There are two knockout groups for this value of \( n \) in the base configuration, with the lower knockout group spanning the upper and lower portions of the configuration.

3. Reflect columns as needed
   For some radices and values of \( n \), we need to reflect all digits in certain
Figure 3: The sequences of codewords for radices (2, 5, 5). (a) The base configuration with ascending and descending sequences in position 0 highlighted in red and blue, respectively. (b) The base configuration with ascending and descending sequences in position 1 highlighted in red and blue, respectively.
Figure 4: The sequence of steps to create a base configuration and knockout groups for radices $(2, 5, 5)$. (a) The reflected cyclic Gray code. (b) The base configuration for radices $(2, 5, 5)$. The dividing line is highlighted in purple and the modular jump is highlighted in orange. (c) The base configuration with knockout groups for $n = 33 = (1, 1, 3)$. (d) The base configuration with knockout groups for $n = 33 = (1, 1, 3)$ and digits in position 0 reflected.
positions before we can stitch the two columns together. As seen in Figure 3, in each column, each digit in each codeword is in an ascending or descending sequence. If the greatest remaining codeword, or $n - 1$, is in the upper portion and any of its digits are in a descending sequence, we must reflect all digits in that position. Similarly, if $n - 1$ is in the lower portion and any of its digits are in an ascending sequence, we must reflect all digits in that position.

In this case, for radices $(2, 5, 5)$ with $n = 33 = (1, 1, 3)$ the largest remaining codeword is 112, which is in a descending sequence in position 0. Codeword 112 is in the upper portion, which means that all digits need to be in ascending sequences, and so we must reflect all digits in position 0. Figure 4(d) shows the base configuration with the knockout group for $n = 33 = (1, 1, 3)$ after we have reflected the digits in position 0. After reflecting position 0, we now have one contiguous knockout group that spans the upper and lower portions. Reflecting position 0 caused the codewords in the lower portion to end with a digit of 4 instead of 0. This movement, however, doesn’t change the position of the modular jump, which remains between the upper and lower portions.

4. **Stitch the subsequent codewords into the code**

The cyclic, dense Gray code is made by using the stitching method for all remaining codewords in the two columns, as seen in Kumar’s thesis. Figure 5(a) illustrates this stitching for $(2, 5, 5)$ with $n = 33 = (1, 1, 3)$. The method involves alternating between the left and right columns in the configuration, connecting codewords that fulfill the Gray-code property. We connect codeword 103 to codeword 104 at the bottom to make this code cyclic. Figure 5(b) has another example of a stitched Gray code for $(2, 5, 5)$ with $n = 37 = (1, 2, 2)$. Note that $n - 1 = 36 = (1, 2, 0)$ is in the lower portion and in all descending sequences, so that no positions were reflected from the base configuration. We connect codeword 101 to codeword 100 on the bottom to make this code cyclic.

Using the stitching method to create our Gray code works for all for radices in this section. We are considering only cases where $n$ is odd, so that there must be an odd number of remaining codewords. Also, the height of each column in the configuration is odd because all radices but the leading radix 2 are odd. All codewords in the left column are remaining since they are less than $n$. Thus, it is guaranteed that there are an even number of remaining codewords in the right column. The even number makes it possible to pair up all remaining codewords in the right column and stitch them into the left column.

Next, we will work through the same steps with radices $(2, 3, 3, 5)$. We see that the method remains the same, and that we are able to create our cyclic, mixed-radix, dense Gray code.

1. **Make the base configuration**
Figure 5: The sequence of steps to create a reflected cyclic Gray code for radices \((2, 5, 5)\). (a) The stitched cyclic, dense Gray code for \(n = 33 = (1, 1, 3)\). (b) The stitched cyclic, dense Gray code for \(n = 37 = (1, 2, 2)\).
(a) **Create the reflected cyclic Gray code**

We create the reflected cyclic Gray code as shown in Figure 6(a).

(b) **Move all codewords ending in 0 to the bottom**

Figure 6(b) shows the base configuration for \((2, 3, 3, 5)\) after all codewords with 0 in position 0 have been moved to the bottom of the configuration in the opposite order that they appear in the reflected Gray code.

2. **Create knockout groups**

Figure 6(c) shows the base configuration after making the knockout groups for \(n = 61 = (1, 1, 0, 1)\) for the radices \((2, 3, 3, 5)\).

3. **Reflect columns as needed**

The codeword 1100 is between two knockout groups at the bottom. There is no way to use the stitching method between the two columns that would allow this codeword to be included. Codeword 1100 is the greatest remaining codeword and is in the lower portion. If any digit in 1100 is in an ascending sequence, we need to reflect all numbers in that digit’s position. As seen in Figure 7(b), 1100 is in an ascending sequence in position 1, and so we must reflect digits in position 1. Figure 6(d) shows the configuration with knockout groups after we have reflected all digits for position 1.

I will provide some intuition to see why 1100 ended up in a position that made it impossible to stitch and why reflecting digits in position 1 fixed that problem. As seen in Figures 7(a) and (b), codeword 1100 is in a descending sequence in position 2 and in an ascending sequence in position 1. The combination of ascending and descending sequences leads to 1100 being the only remaining codeword between knockout groups, as seen in Figure 6(c). Since 1100 is in a descending sequence in position 2, it must be smaller than the codewords directly above it, which all have a value of 2 in this position. Also, because 1100 begins an ascending sequence in position 1, it must be smaller than the codewords below it that increase in position 1. By reflecting all digits in position 1, we flip the ascending sequence to a descending one. Flipping the sequence ensures that 1100 is in descending sequences in both positions 1 and 2, and thus larger than every codeword below it and less than every codeword above. It is now next to the other remaining codewords in the lower portion and can be paired up with them and stitched into the left side of the Gray code.

4. **Stitch the subsequent codewords into the code**

Now the two columns can be stitched together, as shown in Figure 6(e), to create the cyclic, mixed-radix, dense Gray code.

Now that we have seen a few examples of the method to create a cyclic, dense Gray code, I will provide further details on the implementation and justifications
Figure 6: The sequence of steps to create a reflected cyclic Gray code for radices \((2, 3, 3, 5)\). (a) The reflected Gray code. (b) The base configuration. (c) The base configuration with knockout groups for \(n = 61 = (1, 1, 0, 1)\). (d) The base configuration with knockout groups for \(n = 61 = (1, 1, 0, 1)\) and all digits in position 1 reflected. (e) The stitched base configuration with knockout groups for \(n = 61 = (1, 1, 0, 1)\) and all digits in position 1 reflected.
Figure 7: The ascending and descending sequences for radices (2, 3, 3, 5). (a) The ascending and descending sequences in position 2 highlighted in red and blue, respectively. (b) The highlighted sequences for position 1. (c) The highlighted sequences for position 0.
for some steps. We can determine which columns must be reflected in step 3 using only the greatest remaining codeword. I will demonstrate this method for radices (2, 3, 3, 5). Figure 7 shows the direction of the sequences for digits 0, 1, and 2 for these radices.

Observe that, for each portion, every time a sequence changes direction, exactly one digit to the left of the sequence changes by 1. Within a sequence, no digits to the left of the sequence change. For example, consider adjacent codewords 1011 and 1021 in the upper portion in Figure 7(c). Codeword 1011 is in a descending sequence in position 0, and codeword 1021 is in an ascending sequence in position 0. When the direction of the sequence in position 0 changes, the digit in position 1 changes from 1 to 2. This property holds for adjacent codewords 1101 and 1201 as well: they are in descending and ascending sequences in position 0, respectively. When the direction of the sequences changes between these two codewords, the only digit that changes to the left is in position 2. In this case, the digit that changes is in position 2 because the sequences in position 1 are also changing direction between these two codewords. Regardless of where the digit is, it remains that only a single digit changes by exactly 1 every time. This property also holds for the lower portion. Consider codewords 1200 and 1100. As the sequence in position 1 changes from descending to ascending, the digit in position 2 changes from 2 to 1. We also know that all radices are odd except for the leading 2, so that the number of sequences in every position is odd. In our example with (2, 3, 3, 5), there are 1, 3, and 9 sequences in positions 2, 1, and 0, respectively. This property holds for both the upper and lower portions of the configuration, which have the same number of sequences for any given position (except for position 0, where the lower portion does not have directional sequence).

Given that only one digit to the left of a codeword changes when the direction of a sequence changes, we can determine the direction of the sequence by taking the sum of all digits to the left of a position. We create two cases based on whether the greatest remaining codeword is in the upper portion or the lower portion.

**Case 1:** Codeword \( n - 1 \) is in the upper portion.

The first codeword at the top right column of the base configuration is always 10...01. The sum of the digits in positions 1 to 3 for codeword 1001 is \( 1 + 0 + 0 = 1 \). As the sequences in each position change direction, because only one digit to the left changes by 1, the sum of the digits to the left of that position will change parity. We also know that because the first codeword in the upper portion is 1001, and all digits begin in an ascending sequence, that the cumulative sum of digits to the left of a position will be odd if that position is in an ascending sequence. The opposite is also true, so that the cumulative sum of digit to the left of a position will be even if that position is in a descending sequence.

For example, consider codeword 1213. The sum of the digits to the left of position 0 are \( 1 + 2 + 1 = 4 \), which is even. Therefore 1213 must be in a descending sequence in position 0, which we see is true from Figure 7(c).
The sum of the digits to the left of position 1 are $1 + 2 = 3$, which is odd. Therefore 1213 must be in an ascending sequence in position 1, which we see is true from Figure 7(b).

**Case 2:** Codeword $n - 1$ is in the lower portion.

We apply similar logic to the lower portion, where each digit in the first codeword in the lower portion must be in descending sequences. When we moved codewords ending in 0 to the bottom to form the base configuration, we moved them in the opposite order that they appeared in the original reflected Gray code. The greatest codeword ending in 0 was the lowest in the original reflected Gray code, and so it appears first in the lower portion of the base configuration. Additionally, all digits in the greatest codeword in the lower portion are even except for the leading 1 because all radices are odd. These parities mean that the sum of all digits in this codeword must be odd. We see that if the cumulative sum of digits to the left of a position is odd, that position is in a descending sequence, and if the sum is even, that position is in an ascending sequence.

For example, consider codeword 1120. The sum of the digits to the left of position 1 are $1 + 1 = 2$, which is even. Therefore 1120 must be in an ascending sequence in position 1, which we see is true from Figure 7(b). The sum of the digits to the left of position 2 is 1, which is odd, so that 1120 must be in a descending sequence in position 2, which we see is true from Figure 7(a).

When deciding which positions digits should be reflected in, we focus on codeword $n - 1$. If $n - 1$ is in the upper portion, we reflect all digits in positions in a descending sequence. If $n - 1$ is in the lower portion, we reflect all digits in positions in an ascending sequence. These reflections are made to create one contiguous knockout group and put $n - 1$ either just above or just below the knockout group. As illustrated above, positions in a descending sequence in the upper portion have an even cumulative sum, and positions in an ascending sequence in the lower portion have an even cumulative sum. Therefore, we reflect all positions where the cumulative sum of digits to the left is even for $n - 1$.

We call a sequence a **correct sequence** if it is ascending and $n - 1$ is in the upper portion of the configuration, or if it is descending and $n - 1$ is in the lower portion. We are guaranteed that reflecting columns so that $n - 1$ is in all correct sequences will result in a single contiguous knockout group, ensuring our ability to stitch and create a cyclic, mixed-radix, dense Gray code.

To prove the above, we must introduce some new terminology. A **cohort** is a group of $r_0$ codewords that are the same in radices $r_{d-1}, r_{d-2}, \ldots, r_1$, as seen in Figure 8. Exactly one codeword in each cohort must be in the lower portion, and we refer to this codeword as the **leader** of the cohort.

Based on the properties of our base configuration, we are guaranteed some properties about each cohort and their leaders.

**Lemma 4** For any cohort $c_1$ in the upper portion, if that cohort is directly
Figure 8: Examples of cohorts for radices $(2, 5, 5)$. There are $r_0$ codewords in each cohort, and that exactly one codeword in each cohort is in the lower portion. (a) The base configuration with the first cohort highlighted in pink and the second cohort highlighted in orange. (b) The base configuration with all digits in position 0 reflected and the first and second cohort highlighted in pink and orange, respectively.
above cohort \( c_2 \), then the leader of \( c_1 \) will be directly below the leader of \( c_2 \) in the lower portion.

**Proof:** To make the portions of our configuration, we move every codeword ending in a certain digit to the bottom in the opposite order that they appear in the reflected Gray code. Given that all codewords in the same cohort have a unique digit in position 0, exactly one codeword from every cohort moves to the lower portion. Thus, if two cohorts are consecutive, their leaders will also be consecutive in the lower portion. Also, because codewords are moved to the bottom in the opposite order that they appear, the direction of the leaders will be reversed from the direction of the cohorts. 

Lemma 4 is illustrated in Figure 8. We see that the two consecutive cohorts highlighted in pink and orange have consecutive leaders in the lower portion in the opposite order. We see that if a series of cohorts is consecutive in the upper portion, then their leaders must also be consecutive in the lower portion. Next, we examine the properties of each cohort based on the value of \( n - 1 \).

**Lemma 5** If codeword \( n - 1 \) is in all correct sequences and in the upper portion, then all cohorts above it will be remaining and all cohorts below it in the upper portion will be knocked out. If codeword \( n - 1 \) is in all correct sequences and in the lower portion, then all leaders above it in the lower portion will be knocked out and all leaders below it will be remaining.

**Proof:**

**Case 1:** Codeword \( n - 1 \) is in correct sequences in all positions and in the upper portion.
Since codeword \( n - 1 \) is in the upper portion, all of its digits must be in ascending sequences. The only digit that is changing is in position 0 for the cohort \( n - 1 \) is in, which we call \( c_n \). This property means that all codewords in \( c_n \) that are above \( n - 1 \) will be less than \( n - 1 \) and must be remaining. All codewords below \( n - 1 \) in the cohort will be greater than \( n - 1 \) and thus in the knockout group. This property also applies to the cohorts above and below \( c_n \). Because all digits are in ascending sequences, all codewords in cohorts below \( c_n \) must be greater than \( n - 1 \) and knocked out and all codewords in cohorts above \( c_n \) must be less than \( n - 1 \) and therefore remaining.

**Case 2:** Codeword \( n - 1 \) is in correct sequences in all positions and in the lower portion.
Now all digits in \( n - 1 \) must be in descending sequences. Using similar logic as in Case 1, all leaders above \( n - 1 \) must be greater than \( n - 1 \) and thus in a knockout group. Additionally, all leaders below \( n - 1 \) must be less than \( n - 1 \) and thus remaining.

Now we can make a claim about the knockout group for any given \( n \) based on the direction of the sequences.
Theorem 6 For radices \( r = (r_{d-1}, r_{d-2}, \ldots, r_0) \), if all of the digits in positions \( d-2, d-3, \ldots, 0 \) are reflected as necessary so that codeword \( n-1 \) is in all correct sequences, then there is a single contiguous knockout group in the resulting configuration.

Proof: We examine four cases based on whether the greatest remaining codeword is in the upper portion or the lower portion and whether digits in position 0 have been reflected.

Case 1: Codeword \( n-1 \) is in the upper portion and digits in position 0 are not reflected.
We refer to the cohort that \( n-1 \) resides in as cohort \( c_n \). Because position 0 has not been reflected, all codewords in the lower portion end in \( 0 \), so that the leader of cohort \( c_n \) is remaining because it is less than \( n-1 \). As established in Lemma 5, since all digits in \( n-1 \) are in ascending sequences, all codewords above \( n-1 \) in the upper portion are remaining and all codewords below \( n-1 \) in the upper portion are knocked out. This ordering splits the upper portion into two sections: an upper one with remaining codewords and a lower one with codewords in the knockout group.

We also know that the leader for the top-most cohort is the bottom-most codeword in the lower portion because codewords appear in the bottom in the opposite order that they appear in the reflected Gray code, as Figure 8(a) shows. All remaining cohorts in the upper portion must be contiguous, and so by Lemma 4 their leaders must also be remaining and contiguous. All knocked-out cohorts are also contiguous, which means that their leaders must be contiguous and knocked out. This ordering splits the lower portion into two sections: an upper one with knocked out codewords and a lower one with remaining codewords. The knocked out codewords are the lowest section in the upper portion and the upper section in the lower portion, so that they combine to create one contiguous knockout group in the entire configuration.

Case 2: Codeword \( n-1 \) is in the upper portion and digits in position 0 are reflected.
As seen in Case 1, because all digits in codeword \( n-1 \) are in ascending sequences, the upper portion is split into two sections: an upper one with remaining codewords and a lower one with codewords in the knockout group. All digits in the lower portion must end in \( r_0 - 1 \). The greatest remaining leader in the lower portion must be the leader of the previous cohort, or the greatest cohort less than \( c_n \) because the leader of cohort \( c_n \) is greater than \( n-1 \). The previous cohort must be directly above \( c_n \) or completing a descending sequence. This property means that all codewords above the greatest remaining leader in the lower portion will be knocked out and all codewords below the greatest remaining leader will be remaining.
Case 3: Codeword $n - 1$ is in the lower portion and digits in position 0 are not reflected.

All codewords in the lower portion end in 0, so that codeword $n - 1$ must be the only remaining codeword in its cohort. As given by Lemma 5, the lower portion of the configuration has two sections: an upper one with knocked-out leaders and a lower one with remaining leaders. Since $n - 1$ is the topmost remaining leader in the lower section, all other remaining leaders must be contiguous and below it. From Lemma 4, all cohorts in the upper portion for the remaining leaders must be contiguous. This ordering splits the upper portion into an upper section of remaining cohorts and a lower section of knocked out cohorts. The two knocked out sections in the upper and lower portions are next to each other, forming a single, contiguous knockout group.

Case 4: Codeword $n - 1$ is in the lower portion and digits in position 0 are reflected.

All digits in the lower portion must end in $r_0 - 1$, meaning that the leader of each cohort is also the greatest codeword in that cohort. From Lemma 5, the lower portion of the configuration has two sections: an upper one with knocked-out leaders and a lower one with remaining leaders. Because the leader of each cohort is the greatest codeword in that cohort, cohorts are either entirely remaining or entirely knocked out. From Lemma 4, the cohorts in the upper portion must also have two sections: knocked out cohorts or remaining cohorts. Because the lower section of the lower portion consists of remaining leaders, the upper section of the upper portion must be the remaining cohorts. Then, the lower section of the upper portion must comprise contiguous knocked out cohorts. From these properties, we see that there must be one consecutive knockout group between the upper and lower portions. ■

Thus, we have our solution for any number of odd radices with a leading radix of 2. A recap of the process is as follows. First, create the base configuration using the reflected Gray code. Next, create knockout groups for the specified $n$. For every digit in the greatest remaining codeword, if the digit is in a descending sequence and the codeword is in the upper portion, or the digit is in an ascending sequence and the codeword is in the lower portion, reflect all digits in that radix. Then, use the stitching method to create our Gray code.

5 Solution for Leading Radix of 2 With Another Even Radix and Ending with an Odd Radix

Some examples of radices that fall into this category are (2, 4, 5), (2, 3, 5, 4, 7), and (2, 12, 6, 3).

Before moving into how to create the cyclic, dense Gray code, we must first define some new terminology. I will refer to the uppermost cohort in either the
left and right columns of the Gray code configuration as the first chunk. In Figure 9(a), all codewords in the first chunk are highlighted in orange. Note that in this case, the first chunk is in the left column of the configuration, but the first chunk can also be in the right column, as seen in Figure 9(b). All codewords in the first chunk have the same value for all digits, except those in position 0. Figure 9(c) and (d) show the base configuration with every digit in position 0 reflected, but the first chunk remains in the same position.

I will show the process to make this Gray code using example radices (2, 4, 5) and (2, 4, 3, 5). We can split the process into the following steps as seen below. In each step, I provide a brief justification, but we will soon see a more rigorous explanation.

1. Make the base configuration
   (a) Create the reflected cyclic Gray code
       As before, we start with the reflected cyclic Gray code and split this Gray code into two columns: one starting with 0 and one starting with 1. This step is illustrated for (2, 4, 5) in Figure 10(a).
   (b) Move all codewords ending in 0 to the bottom
       Next, take all the codewords that end in 0 and move them to the bottom of each column in the opposite order that they appear in the reflected code as illustrated in Figure 10(b).
   (c) Reflect all sequences in position 0 in the upper portion
       Note that the upper portion consists of an even number—4 in this example—of alternating ascending and descending sequences in position 0. These sequences have been highlighted in Figure 10(c). Then, reflect each of these sequences individually, changing their direction from ascending to descending or vice-versa, as seen in Figure 10(d), to yield our base configuration. Reflecting all sequences in radix 0 ensures that we have a single modular jump in an odd radix. Additionally, this modular jump is between the upper and lower portions, keeping this base configuration consistent with those seen in Section 4.

2. Create knockout groups
   We can then make knockout groups, consisting of all codewords that are greater than or equal to n. Figure 11(a) illustrates the knockout groups for (2, 4, 5) with \( n = 25 = (1,1,0) \).

3. Reflect columns as needed
   We see that codeword \( n - 1 = 24 = (1,0,4) \) is in the upper portion of the configuration. If any digits in the codeword are in descending sequences, we must reflect all digits in those positions. We see that the digit in position 0 is in a descending sequence, which means we must reflect digits in position 0. Figure 11(b) shows the knockout groups after we have reflected digits in this position.
### Figure 9: The first chunk for $\{2, 4, 5\}$.

(a) The base configuration with all codewords in the first chunk highlighted in orange when the first chunk is in the left column.  
(b) The configuration from (a), but with the first chunk in the right column.  
(c) The base configuration with all digits in position 0 reflected and the first chunk highlighted in orange when the chunk is in the left column.  
(d) The configuration from (c), with the first chunk in the right column.
Figure 10: The sequence of steps to create a base configuration for radices (2, 4, 5).

(a) The reflected cyclic Gray code. (b) The reflected cyclic Gray code, with all code-words ending in 0 at the bottom. (c) Part (b) but with ascending and descending sequences in position 0 highlighted in red and blue, respectively. (d) The base configuration, which is part (c) with ascending and descending sequences in the upper portion reflected and highlighted in red and blue, respectively.
Figure 11: The sequence of steps to create a cyclic, dense Gray code for radices $(2, 4, 5)$. (a) The base configuration with knockout groups for $n = 25 = (1, 1, 0)$. (b) The base configuration with knockout groups for $n = 25 = (1, 1, 0)$ and all digits in position 0 reflected. (c) The stitched cyclic, dense Gray code for $n = 25 = (1, 1, 0)$ with the first chunk looped in on the right and highlighted in orange. (d) Another example of the stitched Gray code for $n = 29 = (1, 1, 4)$ now with the first chunk looped in on the left. (e) The stitched cyclic, dense Gray code for $n = 37 = (1, 3, 2)$. 

24
4. **Loop in the first chunk**

We can then loop in the first chunk. To do so, we traverse up the left column and connect to the first chunk when we reach the top. We then traverse down the right column for the entirety of the first chunk. Figure 11(c) shows the resulting configuration with the first chunk highlighted in orange. We connect the 103 and 104 on the right in order to make this code cyclic.

The purpose of looping in the first chunk is to leave an even number of remaining codewords in the right column so that we can pair them and stitch them into the left column of the configuration. The number of codewords in each column is even because there is at least one even radix that is not the leading 2. Additionally, since \( n \) must be odd, the total number of remaining codewords is odd. Since none of the codewords in the left column are in a knockout group, there must be an odd number of remaining codewords in the right column. To end up with an even number of remaining codewords in the right, we must somehow connect up an odd number of remaining codewords. Radix 0 is odd, so that by looping in the first chunk, we connect an odd number of codewords from the right column, leaving an even number of remaining codewords that can be stitched into the left column in the next step.

5. **Stitch the subsequent codewords into the code**

For any subsequent codewords, add them to the Gray code by stitching, as shown in Figure 11(d) for \((2, 4, 5)\) and \( n = 29 = (1, 1, 4) \). The pattern of looping in the first chunk and then stitching in any remaining codewords is repeated for all values of \( n \). Figure 11(e) has another example of a stitched Gray code for \((2, 4, 5)\) with \( n = 37 = (1, 3, 2) \). Note that depending on the value of \( n \) and the parity of codewords above and below the knockout group, which column contains the first chunk can change. We see that in Figure 11(c) the first chunk is in the right column, and in Figures 11(d) and (e) the first chunk is in the left column. We know there must be an odd number of remaining codewords in the right column of our configuration because \( n \) is odd and the height of each column is even. Given that we have one contiguous knockout group, there must either be an odd number of remaining codewords above and an even number below the knockout group, or an even number above and an odd number below. When there is an even number of remaining codewords below the knockout group, the stitching ends at the bottom left, thus the first chunk must be on the left column so we can connect to top and bottom to make our code cyclic. A similar logic applies when there is an odd number of remaining codewords below the knockout group. When the stitching method ends in the right column in the bottom portion, the first chunk must be in the right column.

Next, I will show how to make the cyclic, dense Gray code for \((2, 4, 3, 5)\) by following the same series of steps as seen for radices \((2, 4, 5)\).

1. **Make the base configuration**
(a) **Create the reflected cyclic Gray code**

Starting with the reflected cyclic Gray code, split this Gray code into two columns: one starting with 0 and one starting with 1. This step is illustrated for \((2, 4, 3, 5)\) in Figure 12(a).

(b) **Move all codewords ending in 0 to the bottom**

Next, take all the codewords that have a 0 in position 0 and move them to the bottom of the grid in the opposite order that they appear in the reflected code, as illustrated in Figure 12(b).

(c) **Reflect all sequences in position 0 in the upper portion**

Note that the upper portion consists of alternating ascending and descending sequences in position 0. We reflect each of these sequences so that ascending sequences become descending sequences and vice-versa. This reflection completes our base configuration as seen in Figure 12(c).

2. **Create knockout groups**

We can then make knockout groups, consisting of all codewords that are greater than or equal to \(n\). Figure 13(a) illustrates the knockout groups for \((2, 4, 3, 5)\) with \(n = 77 = (1, 1, 0, 2)\).

3. **Reflect columns as needed**

For some radices and values of \(n\), we need to make changes to the base configuration in order to stitch the two columns together. We see that codewords 1100 and 1101 are both between knockout groups in Figure 13(a). There is no way to use the stitching method between the two columns that would allow us to include these codewords.

The solution to these lone codewords is to reflect all digits for any positions that are not in correct sequences. In this case, because \(n - 1 = 76 = (1, 1, 0, 1)\) is in the upper portion, we must reflect all digits in 1101 that are in descending sequences. Position 1 is in a descending sequence, as seen in Figure 13(b). Figure 13(c) shows the knockout groups after we have reflected all digits for this radix. We see that codewords 1100 and 1101 are no longer between two knockout groups and we have one contiguous knockout group in the entire configuration.

4. **Loop in the first chunk**

We can then make our cyclic, dense Gray code by looping in the first chunk. We connect the 0020 and 0021 on the left in order to make this code cyclic. The first chunk can be seen in orange in Figure 13(d).

5. **Stitch the subsequent codewords into the code**

Next, we stitch the remaining codewords on the right side to create our final Gray code. The stitched Gray code for \(n = 77 = (1, 1, 0, 2)\) is seen in Figure 13(d). Figure 13(e) has another example of a cyclic, dense Gray code for \(n = 101 = (1, 2, 2, 1)\). We connect codewords 1001 and 1000 on the right to make this code cyclic. Note that the column the first chunk is in switches from part (d) to (e) of Figure 13.
Figure 12: The sequence of steps to create a base configuration for radices \((2, 4, 3, 5)\).
(a) The reflected cyclic Gray code. (b) The reflected cyclic Gray code with all codewords ending in 0 moved to the bottom and ascending and descending sequences in position 0 highlighted in red and blue, respectively. (c) The base configuration, created by reflecting all sequences in position 0 for the upper portion. Ascending and descending sequences in position 0 are highlighted in red and blue, respectively.
Figure 13: The sequence of steps to create a cyclic, dense Gray code for radices (2, 4, 3, 5). (a) The base configuration with knockout groups for \( n = 77 = (1, 1, 0, 2) \).  (b) The base configuration shown with ascending and descending sequences in position 1 in red and blue, respectively. (c) The base configuration with position 1 reflected and knockout groups for \( n = 77 = (1, 1, 0, 2) \). (d) The stitched cyclic, dense Gray code for \( n = 77 = (1, 1, 0, 2) \). The first chunk is colored orange. (e) The stitched cyclic, dense Gray code for \( n = 101 = (1, 2, 2, 1) \).
As seen in Section 4, we can determine what direction a sequence is in by taking the cumulative sum of digits to the left of a certain position. In this case, we can use a similar method, but need to make modifications to account for the even radices and reflecting all sequences in the upper portion in step 1(b).

As seen before, every time a sequence changes direction, exactly one digit to the left of the sequence changes by 1. If \( n - 1 \) is in the upper portion we reflect any digits in a descending sequence and if \( n - 1 \) is in the lower portion we reflect any digits in an ascending sequence. Additionally, we can determine whether a digit in \( n - 1 \) is in one of these sequences depending on the parity of the cumulative sum of the digits to the left of a position. When we reflect the digits in an even radix, however, the direction of every sequence to the right of that radix is reflected. For example, take codeword 1321. As seen in Figures 14(b) and (c), 1321 is in a descending sequence in position 1 and an ascending sequence in position 0. Figure 14(d) shows the base configuration with all digits in position 2 reflected. We see that 1321 is in an ascending sequence in position 1 and in a descending sequence in position 0. The direction of these sequences has flipped from Figure 14(b) and (c). After we reflected all digits in position 2, where the radix is 4, all sequences to the right changed direction.

Also, in step 1(b), we reflect all sequences in position 0 in the upper portion. We need to add 1 to the cumulative sum before position 0 to account for this reflection. If the cumulative sum of digits to the left of position 0 after we add 1 is odd, then the codeword is in an ascending sequence in digit 0.

When we reflect an even radix, codewords don’t map to sequences of the same direction. Reflecting all digits in position 2 for \((2, 4, 3, 5)\) means that we swap codewords with 3 and 0 in this position, 2 and 1, and so on. As seen in Figure 14(b), codewords with 0 in position 2 are in an ascending sequence in position 1. The sequence that they are reflected to—codewords with 3 in position 2—are in an ascending sequence in position 1. To account for this change, when we reflect all digits in an even radix, we add 1 to the cumulative sum. Adding 1 changes the parity of all cumulative sums to the right of this position, mirroring how reflecting all digits in an even radix also reflects all sequences to the right.

As seen in Theorem 6, when all digits of \( n - 1 \) are in correct sequences, we must have one contiguous knockout group and be able to stitch to form the Gray code. Because the proof for the theorem doesn’t depend on the radices, it still applies in this section.

Thus, we have our solution for any number of radices with a leading radix of 2 and a rightmost radix that is odd. A recap of the solution is as follows. First, make the base configuration by generating the reflected cyclic Gray code, moving all codewords ending in 0 to the bottom and reflecting all sequences in the rightmost radix in the upper portion. Next, create knockout groups for the specified value of \( n \). For every digit in the greatest remaining codeword, reflect digits in certain columns until all digits in the greatest remaining codeword are in an ascending sequence if the codeword is in the upper portion, or all digits are in descending sequences if the codeword is in the lower portion. Next, loop in the first chunk. Finish by using the stitching method to create our Gray code.
Figure 14: The ascending and descending sequences for radices (2, 4, 3, 5). (a) The ascending and descending sequences in position 2 highlighted in red and blue, respectively. (b) The highlighted sequences for position 1. (c) The highlighted sequences for position 0. (d) The base sequence with all digits in position 2 reflected. The ascending and descending sequences in positions 1 and 0 are highlighted in red and blue, respectively.
Note that this solution requires the first chunk to be complete. I have not yet found a solution where \( n \) is low enough such that codewords in the first chunk are in the knockout group.

6 Solution for Leading Radix of 2 With At Least One Odd Radix and Ending with an Even Radix

Radices that fall into this category include \((2, 5, 6)\) and \((2, 5, 8, 3, 6)\). This case has been solved for radices where \( d = 3 \). I will show the process to make this Gray code using example radices \((2, 5, 6)\). The steps are similar to those for a leading radix of 2 with another even radix and ending in an odd radix as shown in Section 5 above, with a few key differences. We can split the process into the following steps as seen below.

1. Make the base configuration
   
   (a) Create the reflected cyclic Gray code
   
   As before, starting with the reflected cyclic Gray code laid out in Fan’s thesis, we split this code into two columns: one starting with 0 and another starting with 1. This step is illustrated for \((2, 5, 6)\) in Figure 15(a).

   (b) Move all codewords ending in 0 in the rightmost odd radix to the bottom
   
   Next we find the rightmost odd radix, in this case, at position 1. Take all codewords that have a 0 in this position and move them to the bottom of the grid in the opposite order that they appear in the reflected code. This movement splits our grid into an upper portion—where all the codewords have non-zero digits in position 1 and a lower portion—where all codewords have a 0 in that position. This step is illustrated in Figure 15(b). The dividing line between the two portions is drawn in purple.

2. Create knockout groups
   
   After creating the base configuration, make knockout groups by crossing out every codeword that is greater than or equal to \( n \). Figure 15(c) illustrates the knockout groups for \((2, 5, 6)\) with \( n = 43 = (1, 2, 1) \).

3. Reflect columns as needed
   
   For some values of \( n \), it is necessary to reflect all digits in a specific position. To determine whether any columns need to be reflected, we look at the largest remaining codeword and the direction of the sequences it is in. For \( n = 43 = (1, 2, 1) \), the largest remaining codeword is 120 which is between two knockout groups and cannot be stitched into the left column, as seen in Figure 15(c). Codeword 120 is alone because it is in the upper portion and
Figure 15: The sequence of steps to create a base configuration for radices (2, 5, 6).
(a) The reflected cyclic Gray code.  (b) The reflected cyclic Gray code, with all codewords ending in 0 in position 1 moved to the bottom and the dividing line drawn in purple.  (c) The base configuration with knockout groups for $n = 43 = (1, 2, 1)$.  (d) The base configurations with ascending and descending sequences in position 0 highlighted in red and blue, respectively.  (e) The base configuration with knockout groups and position 0 reflected for $n = 43 = (1, 2, 1)$. 

\begin{verbatim}
000 100  015 115  010 110  010 110  015 115
001 101  014 114  011 111  011 111  014 114
002 102  013 113  012 112  012 112  013 113
003 103  012 112  013 113  013 113  012 112
004 104  011 111  014 114  014 114  011 111
005 105  010 110  015 115  015 115  010 110
015 115  020 120  025 125  025 125  020 120
014 114  021 121  024 124  024 124  021 121
013 113  022 122  023 123  023 123  022 122
012 112  023 123  022 122  022 122  023 123
011 111  024 124  021 121  021 121  024 124
010 110  025 125  020 120  020 120  025 125
020 120  035 135  030 130  030 130  035 135
021 121  034 134  031 131  031 131  034 134
022 122  033 133  032 132  032 132  033 133
023 123  032 132  033 133  033 133  032 132
024 124  031 131  034 134  034 134  031 131
025 125  030 130  035 135  035 135  030 130
035 135  040 140  045 145  045 145  040 140
034 134  041 141  044 144  044 144  041 141
033 133  042 142  043 143  043 143  042 142
032 132  043 143  042 142  042 142  043 143
031 131  044 144  041 141  041 141  044 144
030 130  045 145  040 140  040 140  045 145
040 140  005 105  000 100  000 100  005 105
041 141  004 104  001 101  001 101  004 104
042 142  003 103  002 102  002 102  003 103
043 143  002 102  003 103  003 103  002 102
044 144  001 101  004 104  004 104  001 101
045 145  000 100  005 105  005 105  000 100
\end{verbatim}
in a descending sequence in position 0, as seen in Figure 15(d), where all descending sequences in position 0 are highlighted in blue. Reflecting all the digits in position 0 changes this sequence from descending to ascending, placing 120 next to the other remaining codewords in the column, creating one contiguous knockout group as seen in Figure 15(e).

4. Loop in the first chunk

Now we can create the cyclic, dense Gray code by looping in the first chunk. For this section, the first chunk is the bottom-most cohort, along with the first codeword at the top of the configuration. This first chunk is seen highlighted in orange in Figure 16(a). We connect 100 and 110 on the right to make this code cyclic as shown in Figure 16(b). Note that in this case, we loop in 110, along with the first sequence in position 0 at the bottom of the configuration due to the parity of the codewords in each column. As seen in Section 5, the purpose of looping in the first chunk is to leave an even number of remaining codewords in the right column. With an even number of remaining codewords, we are able to pair them and stitch them into the left column of the configuration.

In this case, the radix in position 0 is even, meaning that the column height is also even. We know \( n \) is odd, and so there must be an odd number of remaining codewords in the right column. To end up with an even number of remaining codewords not in the first chunk that we can stitch in, we must loop in an odd number of codewords in the first chunk. Adding 110 to the first chunk ensures that we loop in an odd number of codewords and can stitch the subsequent codewords into the code.

5. Stitch the subsequent codewords into the code

For any subsequent codewords, add them to the Gray code by stitching, as shown for \((2, 5, 6)\) and \( n = 41 = (1, 1, 5) \) in Figure 16(c). Figure 16(d) shows the stitched Gray code after the knockout groups have been made for \( n = 43 = (1, 2, 1) \) and the digits in position 0 have been reflected. Another example of a cyclic, dense Gray code for \((2, 5, 6)\) with \( n = 49 = (1, 3, 1) \) is shown in Figure 16(e). The digits in position 0 have been flipped from part (d) to (e), matching the configuration in part (c). The largest remaining codeword is 130, which is no longer in a descending sequence in position 0; thus we can skip step 3, leaving the base configuration intact before we stitch.

The steps above require that none of the first chunk is in the knockout group. Currently I have not found a solution for values of \( n \) where the first chunk is not yet full. Additionally, a solution for this section where \( d \geq 4 \) has not yet been found. Both of these areas are regions for future work.

7 Conclusion

We now recap the contributions of this thesis:
Figure 16: The sequence of steps to create a cyclic, dense Gray code for radices \((2, 5, 6)\).

(a) All codewords in the first chunk highlighted in orange. In this image, the first chunk is in the right column, but it can be in the left column as well.

(b) The Gray code for \(n = 37 = (1, 1, 1)\) with the first chunk in orange.

(c) The base configuration with knockout groups for \(n = 43 = (1, 2, 1)\).

(d) The stitched Gray code with digit 0 reflected and the first chunk looped in for \(n = 43 = (1, 2, 1)\).

(e) Another example of a stitched Gray code for \(n = 49 = (1, 3, 1)\).
• A new corollary about the number of modular jumps in odd radices required when \( n \) is odd.

• A new corollary about how to create a contiguous knockout group by reflecting columns given the value of \( n - 1 \).

• An algorithm to determine a cyclic, mixed-radix, dense Gray code when \( n \) is odd for a mixed-radix tuple \( r \) where \( r_{d-1} = 2 \) and all other radices are odd.

• An algorithm to determine a cyclic, mixed-radix, dense Gray code when \( n \) is odd and for a mixed-radix tuple \( r \) where \( r_{d-1} = 2 \), \( r_0 \) is odd, and \( n \) falls within \( p_{d-2} + r_0 \leq n < p_{d-1} \).

• An algorithm to determine a cyclic, mixed-radix, dense Gray code when \( n \) is odd for a mixed-radix tuple \( r \) where \( r_{d-1} = 2 \), \( r_0 \) is even, \( d = 3 \), and \( p_{d-2} + r_0 + 1 \leq n < p_{d-1} \).

We have yet to find a comprehensive method to generate a cyclic, mixed-radix, dense Gray code for a mixed-radix tuple \( r \) where there is at least 1 odd radix, \( r_{d-1} = 2 \), \( r_0 \) is even, \( n \) is odd, and \( d > 3 \). Additionally, we have yet to determine a method to generate a cyclic, dense Gray code mixed-radix tuple for \( r \) where there is at least one even radix besides the leading 2 and \( p_{d-2} + 1 < n < p_{d-2} + r_0 \). Finding solutions to both of these cases will be a part of our future work. We also hope to study the generalization of this method to more values of \( n \), as well as radices where \( r_{d-1} > 2 \).

8 Acknowledgements

I would like to thank my thesis advisor, Professor Thomas Cormen of the Dartmouth Computer Science Department (and now also the New Hampshire State Legislature!). Ever since he was my freshman advisor, Professor Cormen has been a mentor and friend to me. It’s because of Professor Cormen that I first got involved with undergraduate research, and I will always be grateful for the guidance, support, opportunities, and encouragement that he provided me with. It’s an honor to have gotten the chance to work and learn from him over the past 4 years.

I would also like to thank the members of my thesis committee, Professor Deeparnab Chakrabarty of the Dartmouth Computer Science Department, and Professor Peter Winkler, of the Dartmouth Computer Science and Mathematics Departments. Their expertise and support helped make this thesis possible.

Finally, I would like to thank the daily 4pm snacks at DALI, Little Sis Nora’s hit single, and all my friends who spent late nights with me, let me teach them binary on a fridge, and heard me talk about this thesis. I truly couldn’t have done this without you.
References


