Gray Codes

University of Liverpool Maths Club, January 2016
Joel Haddley - j.a.haddley@liv.ac.uk

Introduction

Computers count in bits. A bit is a digit that is either 0 (representing 'off') or 1 (representing 'on'). Say we have three bulbs and we want to cycle through all possible on/off states. This is the same as counting from 0 to 7 in base 2:

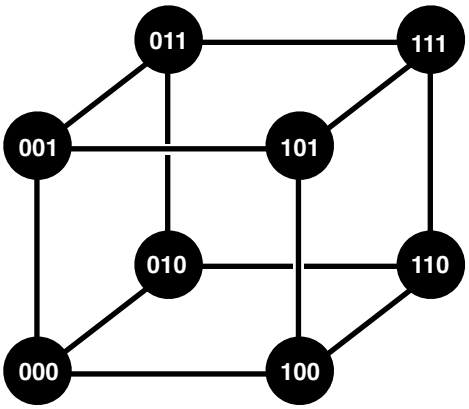
Base 10	0	1	2	3	4	5	6	7
Base 2	000	001	010	011	100	101	110	111

A problem with this strategy is that it is virtually impossible to switch several physical switches at exactly the same time. The transition between the state 011 and 100 requires changing the state of all three switches at once. If you were to try this and film it in slow motion, you would certainly see states other than 011 and 100. Try it!

Gray Codes

In 1947 Frank Gray described a method that solves this problem by introduction an alternative binary notation which became known as Gray Codes.

Base 10	0	1	2	3	4	5	6	7
Gray Code	000	001	011	010	110	111	101	100



This can be visualised as a *Hamiltonian cycle* (look it up!) on the edges and vertices of the 3-dimensional unit cube. So far we've only talked about Gray codes for three bits but Gray codes exist for any number of bits. A 2-bit Gray code is:

Base 10	0	1	2	3
Base 2	00	01	10	11
Gray Code	00	01	11	10

Can you explain that in terms of a 2-dimensional unit cube? What exactly is the 2-dimensional version of a cube? What about higher dimensional cubes?

Reflected Binary Codes

This subheading is the original name that Frank Gray gave to his codes due to the way in which they may be generated. Look at digits 2 and 3 of the 3-digit Gray code. They are exactly the 2-bit Gray codes, followed by the 2-bit Gray codes in reverse. The first half of the of 3-bit Gray code have a leading 0, and the second half have a leading 1. We can generalise this method as follows. Assume that the n-bit Gray codes are given by

$$x_1, x_2, \ldots, x_k,$$

Where $k=2^n$ (why is this?). Then the $(n+1)$-bit Gray codes are given by

$$0x_1, 0x_2, \ldots, 0x_k, 1x_k, \ldots, 1x_2, 1x_1.$$

So n-bit Gray codes exist for any positive integer n.

Further Reading

Gray codes have a real world application in *rotary encoders*. We discussed these in the talk, and there is plenty to read about them online.

Further Problems

Towers of Hanoi:

The standard *n*-bit Gray code gives a solution to the Towers of Hanoi problem with *n* levels. The position of the bit that changes tells us which level of the tower we must move.

Balanced Gray Codes:

Consider again the 2-bit Gray code 00→01→11→10, and count how many times the first digit changes and how many times the last digit changes. Each digit changes twice (note that we're always assuming the final item loops back to the first).

For the 3-bit Gray code (presented above) the 1st, 2nd and 3rd digits change 2, 2 and 4 times respectively.

A Gray code is called *balanced* if each digit changes the same number of times in a full cycle. The 2-bit Gray code we gave is balanced, but the 3-bit Gray code we gave is not. Explain why no balanced 3-bit Gray code exists. Balanced 4-bit Gray codes do exist. Can you find one?

Gray Code Conversion:

Find a rule that converts between an *n*-bit Gray code and its base *n* equivalent, and a rule for going the other way round too (the second of these is not easy to construct but is understandable - you can use the internet to look for such a rule).