Steve Whealton

The Binary Reflected Gray Code

A Gray Code is a system where numbers near one another are represented similarly. More precisely, any two adjacent numbers must have their Gray-Code patterns differ in only one place, and even there it may differ only by one.

There are myriads, even infinities, of possible legitimate Gray Codes. Out of this multitude, one particular realization of the Gray Code criterion has come to be the most important and useful by far. Its proper full name is the "binary reflected gray code" (BRGC). For most people who use it or know about it, it's simply "The Gray Code."

The word, "binary" suggests the fact that The Binary Reflected Gray Code is a system of representing numbers using ones and zeroes, and also that it is a cousin of the system of representing numbers using ones and zeroes that computers regularly utilize. It is a cousin so close that one can easily switch back and forth between regular bit-patterns and brgc bit-patterns for any given integer.

What the brgc transformation does is to take any positive integer and find that number (usually another positive integer) whose **regular** bit-pattern is the **brgc** bit-pattern for the original integer.

Besides meeting The Gray Code criterion, the BRGC has many other useful and beautiful properties. Best of all, it is easily programmed.

Programming the BRGC

Programming the BRGC is almost embarrassingly simple. This stroke of good luck occurs because many programming languages typically have two necessary procedures virtually built right into them.

One is a sort of fixed-point division. In BASIC, this operation has even been given its own symbol, the "\". Thus, "a = b \ c" would mean to divide b by c, to make a the quotient and then to throw away the remainder.

The other operator is designated by "XOR" in BASIC and by "^" in C and in C++. Its name comes from logic, and from the phrase "exclusive or." In English prose, one can think of taking one thing or another, but not both. From this logical beginning, the "XOR" idea

eventually evolved into an operator that can be used between two whole numbers. It considers comparable bits, applying the "one or the other but not both" idea to produce a result. It is equivalent to addition, modulo two.

To find the BRGC Disguise (equivalent) for any given integer, one must first halve the number, throwing away the remainder. Make sure that the original number is also retained. The final step is simply to perform the XOR operation between the original number and the halved-and-trimmed number. The result is the BRGC equivalent.

Using the BRGC

In looking for ways to disguise numbers usefully, one hopes to find a few interesting combinations of regularity and irregularity. With the BRGC, such combinations abound. For one thing, the BRGC equivalent of a number is usually only a little bit smaller or a little bit larger than the number itself. It can never be less than half of the parent number, nor more than double it. Except in the cases of 0 and 1, it can never be the number, itself.

If the BRGC algorithm is applied again and again, the original number eventually returns. The question of how soon it will return leads to perhaps the single most fascinating and useful of all the BRGC's many fascinating and useful qualities.

Since "10" is the brgc bit-pattern for three and "11" is the brgc bit-pattern for two, we would say, using brgc notation, that brgc(2) = 3 and also that brgc(3) = 2. Similarly, it turns out that brgc(0) = 0 and brgc(1) = 1.

Moving higher through the numbers, we discover that brgc(4) = 6, that brgc(5) = 7, that brgc(6) = 5, and that brgc(7) = 4. We also see that 0 and 1 exist in cycles of length one; that 2 and 3 belong to a cycle of length two; and that 4, 5, 6, and 7 belong to a cycle of length four.

It happens that all the numbers between 4 and 15 are members of cycles of size four, while all the numbers between 16 and 255 are members of cycles of size 8.

That is as high as I've investigated exhaustively. But my intuition, along with a few random testings, suggest that for all numbers between 256 and 65535 the cycles are of size 16.

Before listing cycles for numbers larger than 7, let us define a few terms. The *length* of a cycle is one more than the number of transformations necessary before encountering a number already in the given cycle. The *head* of a cycle is the smallest number in it. The *tail* of a cycle is the final number in the cycle begun with the head. Note that the tail is not necessarily the largest number in a cycle, though sometimes it will be. Note also that by necessity brgc(tail) = head, for any cycle.

Let us examine the cycles described already. The length of the first cycle is one. Its head and tail are both zero. In the second cycle, the length is again one, whereas this time the head and tail are both one. In the third cycle, the length is two, the head is two, and the tail

is three. In the fourth cycle, the length is four, the head is four, and the tail is seven. But the tail is seven not because it is the largest number in the cycle, but rather because it is the last number.

On the next pages are listed the cycles for all integers between 0 and 255, inclusive. Each list begins with the cycle's head, and ends with its tail.

Cycles of Length One:

0

1

Cycle of Length Two:

2 3

Cycles of Length Four:

4 6 5 7

8 12 10 15

9 13 11 14

Cycles of Length 8 (Set Ió2 Cycles)

16 24 20 30 17 25 21 31

18 27 22 29 19 26 23 28

Cycles of Length 8 (Set Iló4 Cycles)

32 48 40 60 34 51 42 63

33 49 41 61 35 50 43 62

36 54 45 59 38 53 47 56

37 55 44 58 39 52 46 57

Cycles of Length 8 (Set IIIó4 Cycles)

64 96 80 120 68 102 85 127

65 97 81 121 69 103 84 126

66 99 82 123 70 101 87 124

67 98 83 122 72 100 86 125

Cycles of Length 8 (Set IVó4 Cycles)

72 108 90 119 76 106 95 112

73 109 91 118 77 107 94 113

74 111 88 116 78 105 93 115

75 110 89 117 79 104 92 114

Cycles of Length 8 (Set Vó8 Cycles)

128 192 160 240 136 204 170 255

129 193 161 241 137 205 171 254

130 195 163 243 138 207 168 252

131 194 163 242 139 206 169 253

132 198 165 247 140 202 175 248

133 199 164 246 141 203 174 249

134 197 167 244 142 201 173 251

135 196 166 245 143 200 172 250

Cycles of Length 8 (Set Vló8 Cycles)

144 216 180 238 153 213 191 224

145 217 181 239 152 212 190 225

146 219 182 237 155 214 189 227

147 218 183 236 154 215 188 226

148 222 177 233 157 211 186 231

149 223 176 232 156 210 187 230

150 221 179 234 159 208 184 228

151 220 178 235 158 209 185 229

Many patterns can be discerned by studying these cycles. My favorite conjecture is that these brgc cycles are closely related to Nim Multiplication, and that they can be used in programming Nim Multiplication in a new and more efficient way.

I leave it up to the readers to confirm or reject my conjecture, and to dream up conjectures of their own. I encourage everyone to communicate all and any thoughts and discoveries to me. swhealton@ctsmd.com

Taken together, the diverse features of the BRGC Transformation make it a useful tool. Equally important, when a simple BRGC disguise is used with one or more numbers being fed into a simple formula, the musical or visual result is quite often an interesting one.

Internal Links:

washingtonart.net

HOME

TALK LINKS IMAGES