Gray code

created by lumina

A code for numbers such that the representations of any two successive numbers differ by exactly one digit. The most famous Gray codes are the binary Gray codes, which are used for various applications in logic design and electrical engineering. Compare Karnaugh map, digital logic, Hamiltonian path, hypercube, error-correcting codes, ADC, DAC.

One example of a binary grey code follows. Please note the number of bits that change for each number and the reflections that exist in the counting system.

0 = 0000
1 = 0001
2 = 0011
3 = 0010
4 = 0110
5 = 0111
6 = 0101
7 = 0100
8 = 1100
9 = 1101
10 = 1111
11 = 1110
12 = 1010
13 = 1011
14 = 1001
15 = 1000

The primary use for this is the elimination of race conditions caused by changing two bits at the same time.

Encoding a binary value to a Gray code value is accomplished by the following C function:

unsigned long binary_to_gray(unsigned long n)
Decoding is slightly more tricky, the following function will do it:

```c
unsigned long gray_to_binary(unsigned long n) {
    unsigned long sh;
    unsigned long nn;
    sh = 1;
    nn = (n >> sh);
    while (nn > 0) {
        n ^= nn;
        sh <<= 1;
        nn = (n >> sh);
    }
    return (n);
}
```

It is very interesting to note that binary_to_gray is the same transformation as changing a binary string from fully coherent to differentially coherent (change on 1), and gray_to_binary is the same transformation as changing from differentially-coherent (change on 1) to fully-coherent.

So, to change from binary to Gray: write out the binary string one one line of a page. Copy the left most bit to the line below (assuming the most significant bit is on the left, which it usually is). Repeat for all following bits on the top line: if it's the same as the previous bit, add a '0' to the Gray code; otherwise, add a '1' to the Gray code. (A proof by induction can show that two successive numbers encoded in this manner will have precisely one bit changed after the Gray code transformation.)

To change from Gray to binary: write out the Gray string. Copy the left most bit to the line below. Repeat for all following bits: if it's a '0', copy the previous bit of the binary string; if it's a '1', write the inverse of the previous bit.

The grey code can be padded on the left with 0s just like binary values, or 0s on the left can be ignored just like with binary.

Another interesting property of Gray codes is that (binary_to_gray(x ^ y) == (binary_to_gray(x) ^ binary_to_gray(y)) and (gray_to_binary(x ^ y) == (gray_to_binary(x) ^ gray_to_binary(y))给自己做了一个

Race condition: My brother made $700 in 3 hours by sitting in front of a grocery store
Karnaugh map: hypercube
Hamming code: base 60
Hays Code: Bi-Phase encoding

Printable version

Chaos
Virginity, my loss of
meme virus

Yevgeni Podkletnov
Position Independent Code
right hand path
What Would Jackie Chan Do?

Y'know, if you log in, you can write something here, or contact authors directly on the site. Create a New User if you don't already have an account.