Short Communication / Kurze Mitteilung

An Algorithm for Gray Codes

S. Mossige, Bergen

Received September 22, 1975

Abstract — Zusammenfassung

An Algorithm for Gray Codes. The algorithm generates a list of distinct binary n-tuples such that each n-tuple differs from the one preceding it in just one coordinate [1]. The binary Gray code is often used to generate all subsets of a given set [2]. The whole theory can easily be generalized to generating r-ary codes, $r > 2$, [3].


A Gray code is a function of the numbers $0, 1, ..., 2^n - 1$ onto themselves (i.e. a permutation) such that the images of successive numbers differ at a single-bit position in their binary representation. For example 0, 1, 3, 2 (or in binary 00, 01, 11, 10) is a Gray code for 0, 1, 2, 3. Given a Gray code, the Gray code representation of $N$ is the image of $N$ under the code.

Let $n$ be an arbitrary positive integer. Let $b_1, b_2, ..., b_n$ be the binary digits of a number $N$, $0 \leq N < 2^n$, in standard positional representation and let $c_1, c_2, ..., c_n$ be its representation under the code with "signature" $s_1, s_2, ..., s_n$ where $s_i \in \{-1, 1\}$.

Algorithm A:

A1. Set $b_i \leftarrow 0$ for $i = 1, 2, ..., n$.
A2. Output $b_i$ for $i = 1, 2, ..., n$.
A3. Set $i \leftarrow n$.
A4. Set $b_i \leftarrow b_i + 1$.
A5. If ($b_i = 0$ or $b_i = 1$), go to step A2.
A6. Set $b_i \leftarrow 0$.
A7. ...
A8. Set $i \leftarrow i - 1$.
A9. If $i > 0$, go to step A4.
A10. Terminate.

Algorithm B:

B1. Set $c_i \leftarrow 0$ and $s_i \leftarrow 1$ for $i = 1, 2, ..., n$.
B2. Output $c_i$ for $i = 1, 2, ..., n$.
B3. Set $i \leftarrow n$.
B4. Set $c_i \leftarrow c_i + s_i$.
B5. If ($c_i = 0$ or $c_i = 1$), go to step B2.
B6. Set $c_i \leftarrow c_i - s_i$.
B7. Set $s_i \leftarrow -s_i$.
B8. Set $i \leftarrow i - 1$.
B9. If $i > 0$, go to step B4.
B10. Terminate.
For given $n$ Alg. A generates the binary numbers $0, 1, \ldots, 2^n - 1$ (easily proved by induction on $n$) and as we shall see, Alg. B a Gray code representation of the same numbers.

**Example, $n = 3$**

<table>
<thead>
<tr>
<th>$b_1$</th>
<th>$b_2$</th>
<th>$b_3$</th>
<th>$-c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
<th>$s_1$</th>
<th>$s_2$</th>
<th>$s_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

In ordinary binary addition the digit $b_i$ is changed every $2^{n-i}$ step. In Alg. B we put $s_i = -s_i$ but do not change $c_i$ when we have a carry in position $i$, which means that after the first $2^{n-i}$ steps, $c_i$ is changed every $2^{n-i+1}$ step. As we shall see, what the signature actually does is to determine how far left the carry propagates with each addition of 1 when counting in base 2.

*The statements in Alg. A and Alg. B are executed in exactly the same sequence from initialization until termination.*

This is proved by induction on $n$. It is easily verified for $n = 1$. Let us assume that the above proposition is true for $n - 1$. The crucial statement is step 5. Let us start at step 2 with $b_1, b_2, \ldots, b_n$ and $c_1, c_2, \ldots, c_n$. Then, because of the assumption the exit from step 5 will be the same in both algorithms for $i = n, n - 1, \ldots, 2$. The initial values of $b_1, c_1$ and $s_1$ are 0, 0 and 1 respectively. Using the algorithms it is easily verified that for $i = 1$ we always have the same exit from step 5 because of periodicity in the variation of the values of $b_i, c_i$ and $s_i$. Since Alg. A generates the binary numbers $0, 1, \ldots, 2^n - 1$, both algorithms terminate after $i = 1$ the second time.

*Alg. B generates a Gray code of length $2^n$.*

First, in Alg. B one and only one $c_i$ is changed between each execution of step B2.

The relation between $b_1, b_2, \ldots, b_n$ and $c_1, c_2, \ldots, c_n$ is the **conversion rule**

$$
c_i = b_{i-1} \ XOR \ b_i
$$

with

$$
s_i = \begin{cases} 
-1 & \text{if } b_{i-1} = 1 \\
1 & \text{otherwise}
\end{cases}
$$

for $i = 1, 2, \ldots, n$ and $b_0 = 0$. The inverse of this is seen to be

$$
b_i = b_{i-1} \ XOR \ c_i \quad \text{or} \quad b_i = 1 - c_i \quad \text{if } s_i = -1, \text{otherwise } b_i = c_i
$$