Gray Codes and Hamiltonian Circuits on the n-Cube

Bit Strings and Gray Codes

For any positive integer $n$, there are exactly $2^n$ bit strings of length $n$.

For $n = 1$ we list the possible strings as 0 1.

For $n = 2$, they are (in the usual order)

\[
00 \quad 01 \quad 10 \quad 11
\]

We want to reorder these so that successive bit strings differ in exactly one bit. This can be done as follows:

\[
00 \quad 01 \quad 11 \quad 10
\]

Successive entries in the list differ in one position (including the last entry and the first entry). This ordering of the bit strings of length 2 is called a 2-bit Gray code.

This 2-bit code can be obtained from the 1-bit code by a “reflection process”.

List the 1-bit code followed by a second “reflected” copy:

\[
0 \quad 1 \quad 1 \quad 0
\]

Then, prepend a 0 to the first set of codes and a 1 to the reflected set, obtaining.

\[
00 \quad 01 \quad 11 \quad 10
\]
We can construct a 3-bit Gray code from the 2-bit Gray code given above as follows:

List the 2-bit codes, followed by the same codes in reverse order. Then put a 0 in front of the first 4 codes and 1 in front of the second 4. Thus

\[
\begin{array}{c}
00 & 000 \\
01 & 001 \\
11 & 011 \\
10 & 010 \\
10 & 110 \\
11 & 111 \\
01 & 101 \\
00 & 100
\end{array}
\]

The second column then gives a 3-bit reflected Gray code. The word ‘reflected’ refers to the fact that this 3-bit code was constructed from the 2-bit code, by listing the 2-bit followed by the “reflected” version of that code (that is, the code in reverse order) and then prepending the 0s and 1s as shown.

In general, an \( n \)-bit reflected Gray code can be constructed recursively as follows:

Case \( n = 1 \): A 1-bit Gray code is \( 0 \ 1 \).

Case \( n > 1 \): Assuming that we know how to construct an \((n-1)\)-bit Gray code, construct an \( n \)-bit Gray code by:

- List the \((n-1)\)-bit Gray code, then list it again in reverse order.
- In front of the terms of the \((n-1)\)-bit Gray code put a 0 and in front of the second ‘reflected’ set of terms, put a 1.

The resulting list of \((n+1)\)-bit codes is an \((n+1)\)-bit Gray code.

**The Gray Code and Hamiltonian circuits on the n-Cube**

The graph called the \( n \)-cube can be described as follows: Use as vertices the bit strings of length \( n \) (there are \( 2^n \) of them, so the \( n \)-cube has \( 2^n \) vertices). Connect two vertices by an edge exactly when the two bit strings differ in exactly one bit. The recursive Gray code construction given above leads to a natural way of constructing a Hamiltonian circuit on an \( n \)-cube graph for any \( n \geq 2 \): Just follow the Gray code ordering of the vertices of the \( n \)-cube.
\( n = 1 \): There is no Hamiltonian circuit on the 1-cube since if we go from 0 to 1, then we can’t return to 0 without reusing the edge from 0 to 1.

\( n = 2 \): Just as in the construction of the 2-bit Gray code, we make two copies of the 1-cube, label the first using the Gray code 0 1, with a 0 prepended to each. Thus the vertices are 00 01. Label the second copy the same way, but prepending a 1 on each. Thus the vertices are 10 11. Then put in edges between vertices differing in exactly one bit. This results in a 2-cube and a Hamiltonian circuit is described by following the order of the Gray code: 00-01-11-10-00

\( n = 3 \): We make two copies of the 2-cube, labeling each with the 2-bit strings as above. We prepend a 0 to each string on the first copy and a 1 to each string on the second copy. Then put in edges between vertices differing in exactly one bit. This results in a 3-cube and a Hamiltonian circuit is described by following the order of the Gray code:

000-001-011-010-110-111-101-100-000
We make two copies of the 3-cube, labeling each with the 3-bit strings as above. We prepend a 0 to each string on the first copy and a 1 to each string on the second copy. Then put in edges between vertices differing in exactly one bit. This results in a 4-cube and a Hamiltonian circuit is described by following the order of the Gray code:

0000-0001-0011-0010-0110-0111-0101-0100
-1100-1101-1111-1110-1010-1011-1001-1000

Exercises

1. Draw a picture of the 4-cube.
2. How many edges do the 1-cube, 2-cube, and 3-cube have?
3. How many edges does the $n$-cube have? Let $E_n$ denote the number of edges of the $n$-cube graph. Then from the above examples, we have $E_1 = 1, E_2 = 4, E_3 = 12$. To find the general formula for $E_n$, find a recursion relating $E_{n+1}$ and $E_n$. Using an appropriate initial condition, solve the recursion using the methods developed in this course, to obtain the formula for $E_n$ as a function of $n$. 

$n = 4$: We make two copies of the 3-cube, labeling each with the 3-bit strings as above.