7-2008

De Bruijn Cycles

Val Adams
University of Nebraska - Lincoln

De Bruijn Cycles

Val Adams

In partial fulfillment of the requirements for the Master of Arts in Teaching with a Specialization in the Teaching of Middle Level Mathematics in the Department of Mathematics.
David Fowler, Advisor

July 2008

Introduction

Problem 1. A robot is moving on a cyclic track. The track is marked at evenly spaced intervals with 0s and 1s, with a total of 8 marks. The robot can see the 3 marks closest to him. How should the 0s and 1s be put on the track so that the robot knows where on the track he is by just looking at the 3 closest marks?

Problem 2. The city of Konigsberg, Prussia is set on the Pregel River and includes two large islands, which are connected to each other and the mainland by seven bridges. Is it possible to walk a route that crosses each bridge exactly once?

Problem 3. Sally wants to make a necklace. She has several colors of beads to make her necklace. She wants to make a necklace in which there are no repeating patterns. How should she arrange her beads on the string?

The problems above seem to be very different, but mathematically they are the same. All of them can be solved using De Bruijn Cycles.

Nicolaas Govert de Bruijn

Nicolass Govert De Bruijn, known for his De Bruijn Cycles, was born on July 9, 1918. De Bruijn is a Dutch mathematician affiliated with Eindhoven University of Technology. While at Eindoven University, he has studied many math concepts. Currently he is studying the brain model.

De Bruijn Cycles

Each cycle for the problems mentioned in the introduction can be constructed using a De Bruijn cycle. A De Bruijn cycle is a cyclic string of an alphabet of length *n* such that all of the substrings of length *k* consecutive symbols are distinct. The alphabet is a collection of symbols: for example $\{x, y, z\}, \{r, g, b\}, \{0,1\}, \{a, b, c, ..., z\}$. For the purposes of this paper, I will be using a two character alphabet $\{0,1\}$. The characters do not represent numerical values where they can have mathematical operations performed on them, so they can not be added, subtracted, and so on.

Now recall that in the robot problem, a robot is moving on a cyclic track. The track is marked at evenly spaced intervals with 0s and 1s, with a total of 8 marks. The robot can see the 3 marks closest to him. Here is what the track should look like:

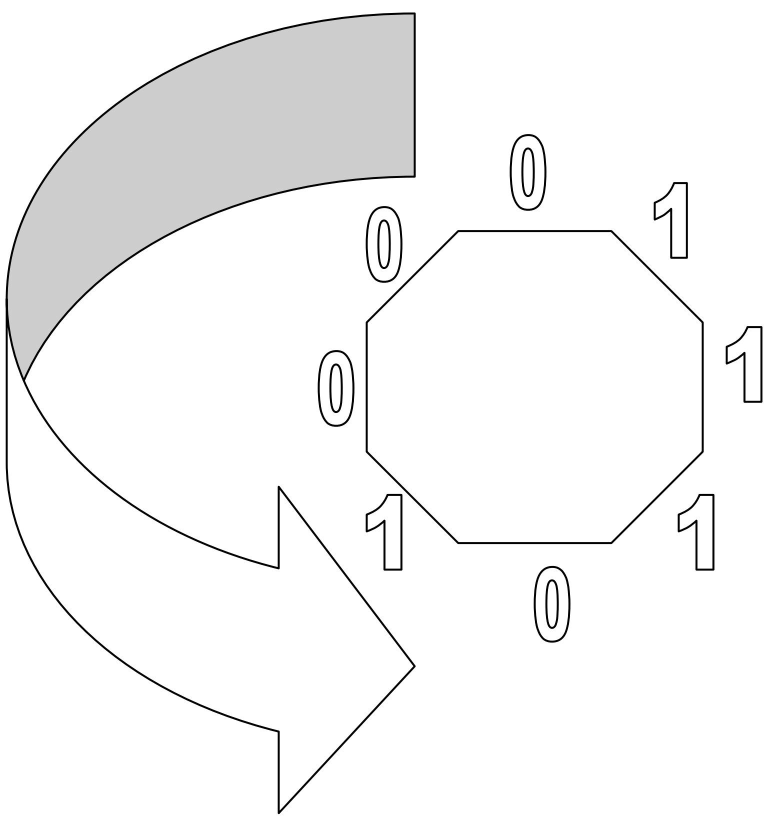


Figure 1:De Bruijn Cycle where *n*=2 and *k*=3

From this figure we can see that a De Bruijn cycle could be 00010111 or 11000101,or another equivalent string depending on the starting point. We would say that the cycle length is 8 since it can be identified by a string of eight characters. I will use this cycle as my standard example. A systematic method to find the cycles will be shown later in the paper. Observe that at any point on the track the robot can locate where he is by the closest three marks. This robot cycle represents a De Bruijn cycle when n=2 and k=3.

If we change the scenario so that the robot can see only the closest 2 marks, the result is a much simpler cycle. There is now a two character alphabet, a substring length of 2, and a cycle of length 4.

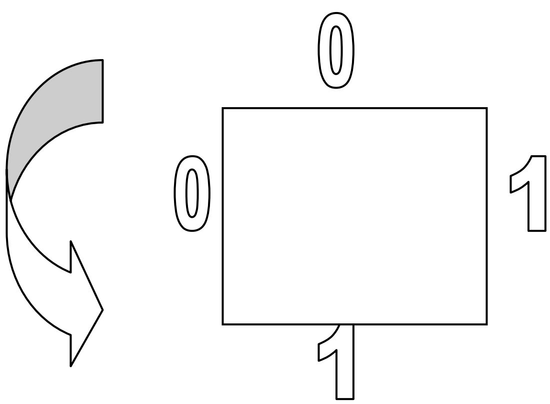


Figure 2: De Bruijn Cycle where *n*=2 and *k*=2

Here is another De Bruijn cycle where n=8 and k=3, which I found using the De Bruijn Applet, www.hakank.org/comb/deBruijnApplet.html .

0001002003004005006007011012013014015016017021022023024025026027031032033034035036037041042043044045046047051052053054055056061062063064065066067071072073074075076077111211311411511611712012112212312412512612713013113213313413 5…6676777(000)

Notice that this cycle length is much longer, and the alphabet is much longer.

For the standard example, the number of ways to arrange 2 characters in a string of length 3 is 2^3, so there are 2^3=8 different sets of three.

000
001
011
111
110
101
010
100
(000)

Figure 3: Showing the overlap in the cycle

Notice that when arranging the sets of characters in this way, each set of three characters repeats the last two characters of the previous set of three. Similar arrangements of other De Bruijn cycles must always display this overlap. After using the De Bruijn Applet site to create several cycles with various numbers of characters and substring lengths, I found that the length of the cycles or the number of distinct substrings of a particular length, *k*, could be determined by n^k. Going back to the robot problem where *n*=2 and *k*=3, the length of the cycle or number of substrings of length 3 is $2^3 = 8$. The cycle length and number of substrings in the sequence when *n*=2 and *k*=2 is $2^2 = 4$. The cycle length and number of substrings of length 3 in the sequence when *n*=8 and *k*=3 is $8^3 = 512$. Using a table I can see the possible values for *n*, *k*, and n^k.

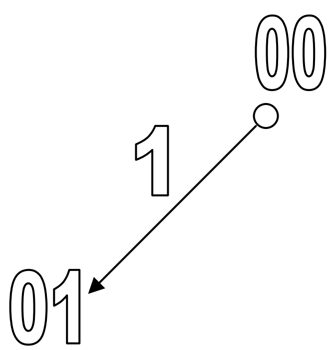
n	k	n^k
0	0	1
1	0	1
1	1	1
1	4	1
2	0	1
2	1	2
2	2	4
2	7	128
3	0	1
3	1	3
3	2	9
15	0	1
15	1	15
15	2	225

Figure 4: Table showing values of *n*, *k*, n^k

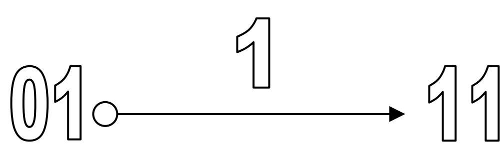
Notice that while *n* and *k* can be any pair of positive integers, some examples are worth more attention than others. As indicated by the table, if $k = 0$ there is only one choice for the cycle no matter how many characters are in the alphabet. If the alphabet is 1 character, the number of substrings or length will always be 1. The table entries for n and k are most useful when they are both greater than or equal to 2.

A De Bruijn cycle can be represented by a directed graph, also known as a digraph. Revisiting the robot problem, we can use a digraph to represent a De Bruijn cycle and obtain information about the number of substrings in the cycle where n=2 and k=3. We can follow the digraph to get a cycle with 8 sets of 3 characters that do not repeat. To do this I will start at 00 and add a 1.

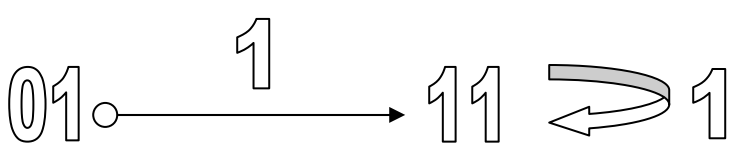
Think of sliding this 1 from the right hand side of the two digits, replacing the left-most digit.



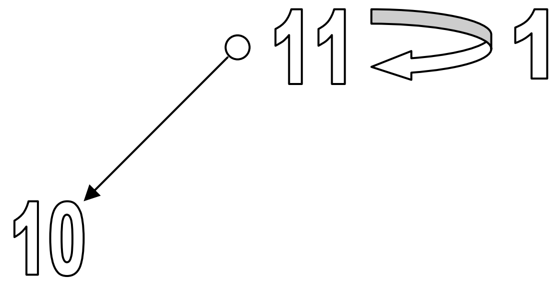
Then from 01 add a 1.



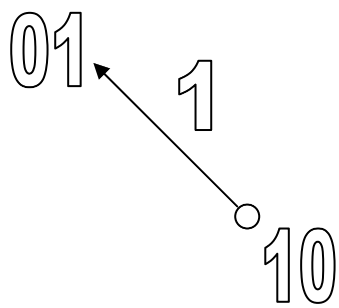
Repeat the adding of a 1, to get another 11.



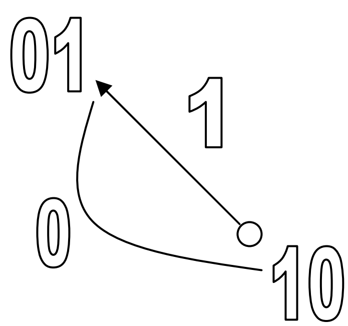
From 11 add a 0.



From 10 add a 1.



Then adding a 0, to return back to 10.



Then the final move returns to 00, by adding a 0.

Placing the steps all together we get the following digraph:

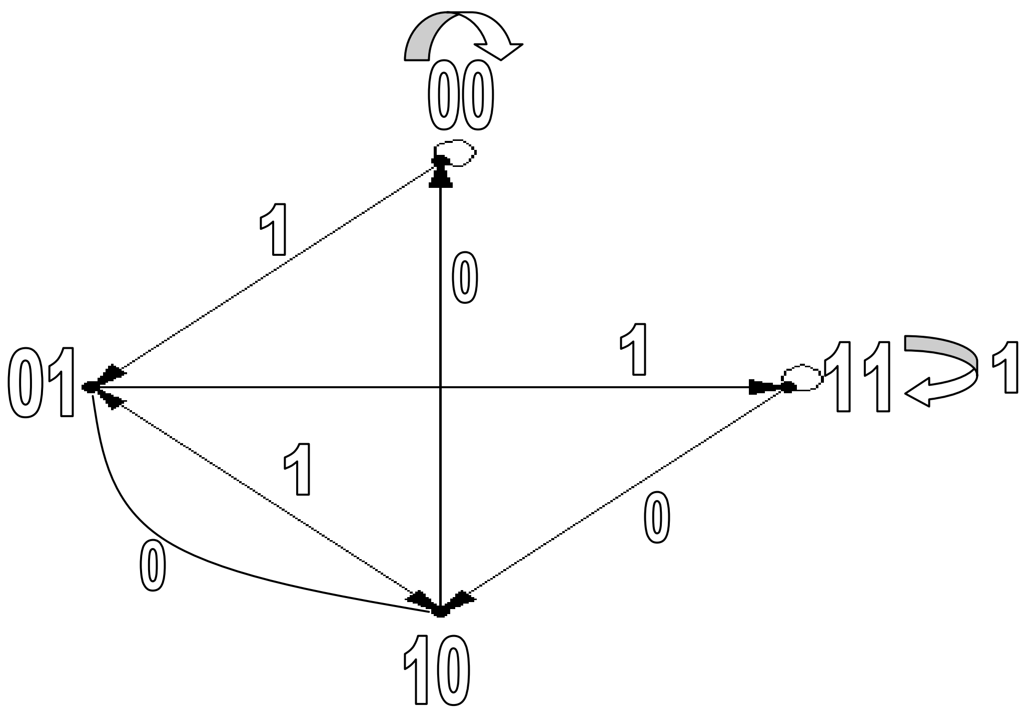


Figure 5:Digraph of Robot Problem

This matches the order of the cycle (00101110) obtained previously for the robot problem starting with characters 00. This will also show up in the shift register example found in a later section.

<u>The Bridges of Konigsberg</u>

The Bridges of Kongsberg, a well-known math problem, is created around an actual place and situation. The city of Konigsberg is located in Russia on the Pregel River. There are two large islands in the city, which were connected to the mainland area with seven bridges. The problem was to decide if there was a possible route to walk over all of the bridges exactly one time. Leonhard Euler in 1736 proved that it was not possible. He showed this by replacing each landmass with a dot, which he called a vertex, and   each bridge with a line, which he called an edge.

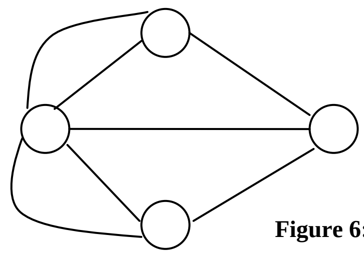


Figure 6:Graph of Euler's Seven Bridges Problem

Then by examining all possible routes traveling along the edges, he found that no matter which vertex he started from there was no route that would travel over each bridge exactly one time.

Euler found that the solution to this problem could be explained using the degrees of the vertices. A degree of a vertex is determined by the number of edges that extend from it. Euler observed that all of the vertices in the Seven Bridges Problem were of odd degree, meaning that there were an odd number of edges extending from each vertex. If there is an odd degree at a vertex, at some point once the vertex has been left, there will be no edge that has not been traveled on which to return. Euler proved that a path traversing all edges could be made if there were two or zero vertices with odd degree. He also found that if there were two vertices with odd degree, they must be the starting and ending points of the path. Since in the Seven Bridges Problem the four vertices are all of odd degree there is no such path. The path in which all bridges could be traveled exactly once became known as the Euler path.

Eulerian Circuits

A *circuit* is a path that begins and ends at the same vertex. An *Eulerian path* is a path that visits each edge of a graph exactly once. An *Eulerian circuit* is a path that begins and ends at the same vertex and also visits each edge exactly once. All vertices on a graph that contains an Eulerian circuit are of even degree, meaning there are an even number of edges stemming from each vertex. An example referred to as the beaded necklace problem can be represented by the graph below. In this problem the bead color order is being determined for the case that includes 5 different colors (pink, red, yellow, green, and blue) and substring length of 1.

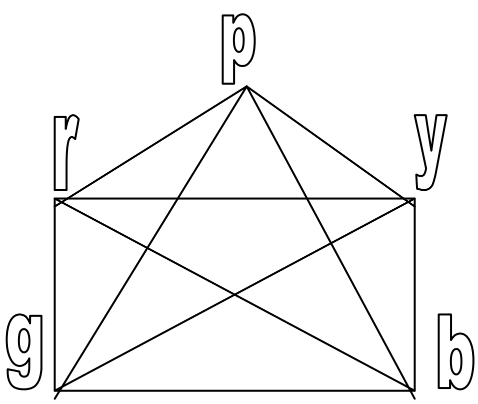


Figure 7: A digraph for the Beaded Necklace problem

The graph has an Eulerian circuit because each vertex has an even degree. An example of a cycle in the circuit is p, r, y, g, b. This also represents a De Bruijn cycle.

<u>De Bruijn digraphs, De Bruijn cycles, and Eulerian circuits</u>

The De Bruijn digraph is a picture of many De Bruijn Cycles. To determine a De Bruijn Cycle from the digraph, one begins by choosing a vertex, then follows the edges in the direction of the arrows. This was demonstrated previously in Figure 5, which is displayed again here for easy reference.

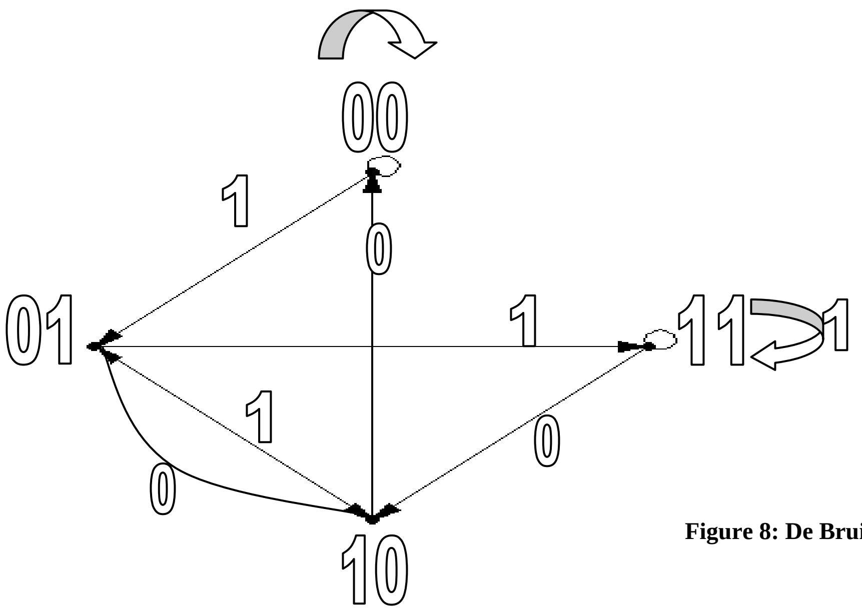


Figure 8: De Bruijn Digraph where *n*=2

This next example is a digraph where *n*=2 and *k*=4. The number of vertices and edges have increased as the substring has been made longer (http://en.wikipedia.org/wiki/De_Bruijn_sequence).

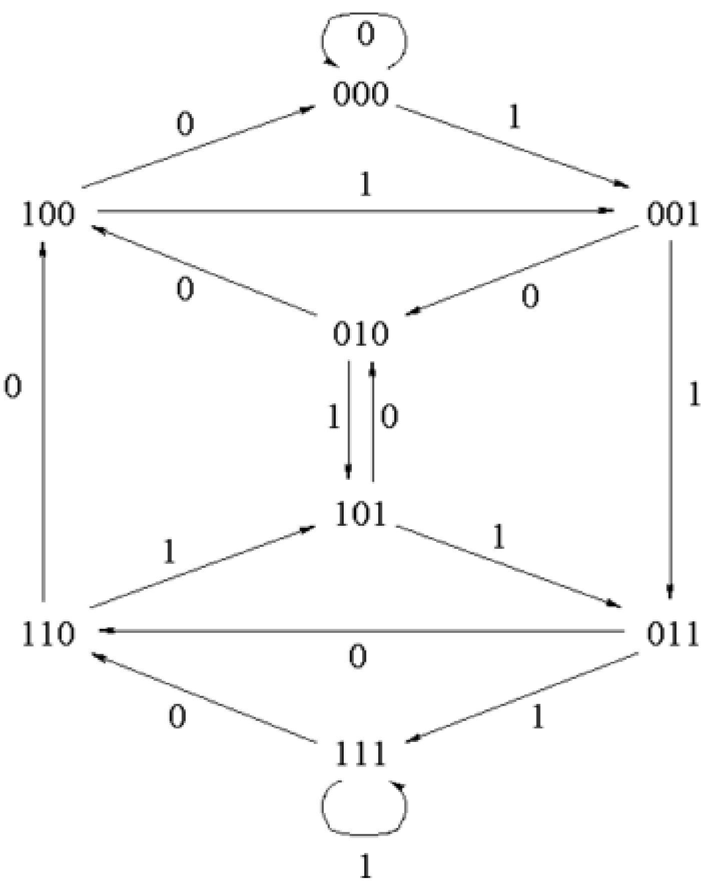


Figure 9:De Bruijn digraph where *n*=2 and *k*=4

It can be observed that the De Bruijn digraph is Eulerian. Since all vertices are of even degree it contains an Eulerian circuit..

An Eulerian circuit can be used to construct a De Bruijn cycle by using the De Brujin digraph. For example in figure 8, starting with the vertex 110, going up to 100, right to 001, left to 010, down to 101, right to 011 and back to 110 (no additional character is added for the last move, it shows that the cycle is cyclic). This creates the De Bruijn cycle that looks like the following:

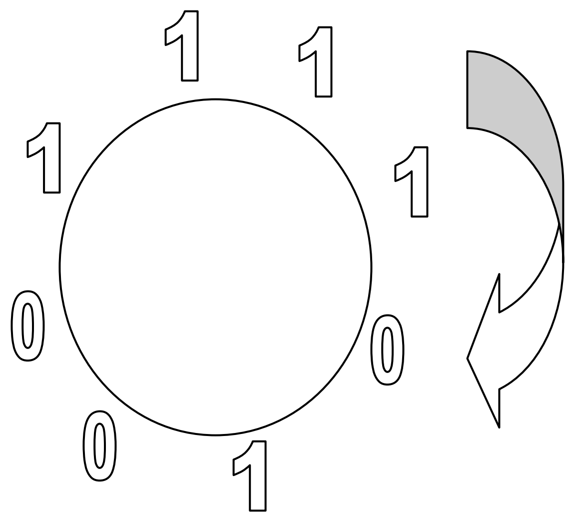


Figure 10: De Bruijn cycle created from Eulerian curcuit

This shows that a De Bruijn cycle can be made using the digraph and an Eulerian cicuit.

<u>Generalizations and Further Questions</u>

When a larger alphabet is used, the length of the cycle increases significantly. Let's compare it to the robot problem, when the alphabet had 2 characters and the robot could only see 3 marks at a time, so n=2 and k=3, there were a total of 8 marks. If the alphabet is increase to 3 characters, say $\{0,1,2\}$, so that n=3 and k=3, there are now 27 marks. If the alphabet is now 4, where n=4 and k=3, there are now 64 marks. The number of marks represent the number of substrings of the specified length. If the alphabet was length q, where n=q with substrings length k, there would be a total of q^k marks or q^k substrings length k. The length of any De Bruijn cycle is q^k, where q is the number of letters in the alphabet and k is the length of the substring.

De Bruijn cycles have many applications in other fields. Two examples that will be considered here are shift registers and DNA sequencing. Shift registers are used for storing information in electronics. A computer shift register stores a set of bits (0s and 1s) in a particular order. When a new bit of information is entered in, it is added at the end of the register. Then a bit is "shifted off" the other end. For example let's look at a three bit register:

Start with {0, 0, 0} and then add a 1 from the right. The left-most zero will be bumped. Then add a 0 to the right and the left-most 0 will be bumped. Then add a 1 to the right and the left-most 0 will be bumped.

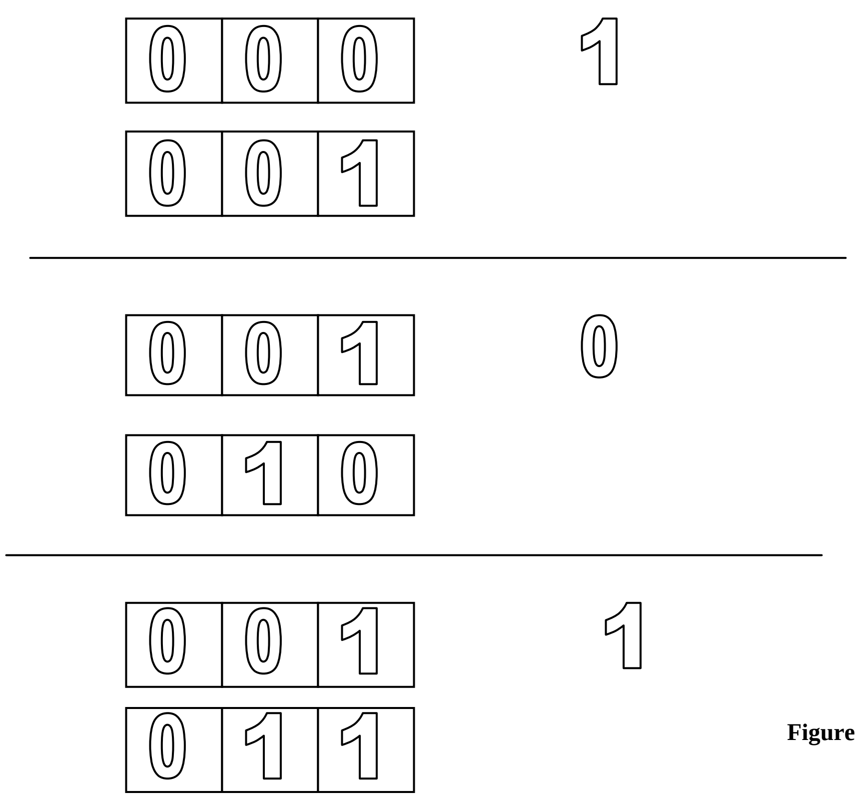


Figure 11: Example of shift register

This looks like the digraph of the De Bruijn cycles in figure 9, as you move from a vertex along an edge the left-most character is bumped.

We now consider Deoxyribonucleic acid (DNA) sequencing. DNA sequences have a four character alphabet: adenine (A), thymine (T), guanine (G), and cytosine (C). DNA is connected with bonds that are called base pairs, which are made up of A, T, G, and C. DNA does have very specific things that happen; for example, T pairs with A and G pairs with C. An example of a DNA sequence that shows a pair of double-strand patterns:

ATCGAT
TAGCTA

Figure 12: DNA sequence

Notice that the top sequence starts with an A and ends with a T while the bottom sequence does the exact opposite. This is because A pairs with T and G pairs with C. The sequence then cycles in a manner very similar to the De Bruijn sequences noted previously, the only difference being the alphabet length. I found many distinct sequences with DNA, each fit very specific characteristics, some of which were very complicated. There is much more that can be said about the sequencing with DNA that will not be addressed in this paper.

Classroom Activity

I have found that there are many applications for Eulerian circuits and De Bruijn cycles. The overall idea of the application of De Bruijn cycles is to identify a cyclic pattern that does not repeat. This could involve the beading of a necklace, marks on a track giving a robot information, or routes over bridges. Each seems to be so different, but all can be explained using De Bruijn cycles. As I researched for classroom activities, I found there are many activities that can be done with all ages of students in varied degrees of difficulty.

References

Wheeler, E. & Brawner, J. (2005). *Discrete Mathematics for Teachers*. Boston, MA: Houghton Mifflin Company.

Websites:

http://en.wikipedia.org/wiki/Base_pair

http://en.wikipedia.org/wiki/Bridges_of_Konigsberg

http://en.wikipedia.org/wiki/De_Bruijn_sequence

http://en.wikipedia.org/wiki/Bridges_of_Konigsberg

http://en.wikipedia.org/wiki/Nicolaas_Govert_de_Bruijn

http://learn.genetics.utah.edu/units/basics/builddna/

http://seqcore.brcf.med.umich.edu/doc/educ/dnapr/sequencing.html

http://www.hakank.org/comb/deBruijnApplet.html