De Bruijn Sequences

Prepared by Mark on February 14, 2025
Based on a handout by Glenn Sun

Instructor's Handout

This file contains solutions and notes.
Compile with the "nosolutions" flag before distributing.
Click [here] for the latest version of this handout.

Part 1: Introduction

Example 1:
A certain electronic lock has two buttons: `0` and `1`. It opens as soon as the correct two-digit code is entered, completely ignoring previous inputs. For example, if the correct code is 10, the lock will open once the sequence `010` is entered.

Naturally, there are $2^2 = 4$ possible combinations that open this lock.
If we don't know the lock's combination, we could try to guess it by trying all four combinations.
This would require eight key presses: `0001101100`.

Problem 2:
There is, of course, a better way.
Unlock this lock with only 5 keypresses.

Solution

The sequence `00110` is guaranteed to unlock this lock.

Now, consider the same lock, now set with a three-digit binary code.

Problem 3:
How many codes are possible?

Problem 4:
Show that there is no solution with fewer than three keypresses

Problem 5:
What is the shortest sequence that is guaranteed to unlock the lock?
Hint: You'll need 10 digits.

Solution

`0001110100` will do.

Part 2: Words

Definition 6:
An *alphabet* is a set of symbols.
For example, $\{\texttt{0}, \texttt{1}\}$ is an alphabet of two symbols, and $\{\texttt{a}, \texttt{b}, \texttt{c}\}$ is an alphabet of three.

Definition 7:
A *word* over an alphabet A is a sequence of symbols in that alphabet.
For example, `00110` is a word over the alphabet $\{\texttt{0}, \texttt{1}\}$.
We'll let $\varnothing$ denote the empty word, which is a valid word over any alphabet.

Definition 8:
Let v and w be words over the same alphabet.
We say v is a *subword* of w if v is contained in w.
In other words, v is a subword of w if we can construct v
by removing a few characters from the start and end of w.
For example, `11` is a subword of `011`, but `00` is not.

Definition 9:
Recall Example 1. Let's generalize this to the *n-subword problem*:
Given an alphabet A and a positive integer n, we want a word over A that contains all possible length-n subwords. The shortest word that solves a given n-subword problem is called the *optimal solution*.

Problem 10:
List all subwords of `110`.
Hint: There are six.

Solution

They are $\varnothing$, `0`, `1`, `10`, `11`, and `110`.

Definition 11:
Let $\mathcal{S}_n(w)$ be the number of subwords of length n in a word w.

Problem 12:
Find the following:
- $\mathcal{S}_n(\texttt{101001})$ for $n \in \{0, 1, ..., 6\}$
- $\mathcal{S}_n(\texttt{abccac})$ for $n \in \{0, 1, ..., 6\}$

Solution

In order from $\mathcal{S}_0$ to $\mathcal{S}_6$:
- 1, 2, 3, 4, 3, 2, 1
- 1, 3, 5, 4, 3, 2, 1

Problem 13:
Let w be a word over an alphabet of size k.
Prove the following:

- $\mathcal{S}_n(w) \leq k^n$
- $\mathcal{S}_n(w) \geq \mathcal{S}_{n-1}(w) - 1$
- $\mathcal{S}_n(w) \leq k \times \mathcal{S}_{n-1}(w)$

Solution

- There are k choices for each of n letters in the subword. So, there are k^n possible words of length n, and $\mathcal{S}_n(w) \leq k^n$.
- For almost every distinct subword counted by $\mathcal{S}_{n-1}$, concatenating the next letter creates a distinct length n subword. The only exception is the last subword with length $n-1$, so $\mathcal{S}_n(w) \geq \mathcal{S}_{n-1}(w) - 1$
- For each subword counted by $\mathcal{S}_{n-1}$, there are k possibilities for the letter that follows in w. Each element in the count $\mathcal{S}_n$ comes from one of k different length n words starting with an element counted by $\mathcal{S}_{n-1}$. Thus, $\mathcal{S}_n(w) \leq k \times \mathcal{S}_{n-1}(w)$

Definition 14:
Let v and w be words over the same alphabet.
The word vw is the word formed by writing v after w.
For example, if $v = \texttt{1001}$ and $w = \texttt{10}$, vw is $\texttt{100110}$.

Problem 15:
Let F_k denote the word over the alphabet $\{\texttt{0}, \texttt{1}\}$ obtained from the following relation:

$$F_0 = \texttt{0}; \quad F_1 = \texttt{1}; \quad F_k = F_{k-1}F_{k-2}$$

We'll call this the *Fibonacci word* of order k.

- What are F_3, F_4, and F_5?
- Compute $\mathcal{S}_0$ through $\mathcal{S}_5$ for F_5.
- Show that the length of F_k is the $(k+2)^{\text{th}}$ Fibonacci number.
  Hint: Induction.

Solution

- $F_3 = \texttt{101}$
- $F_4 = \texttt{10110}$
- $F_5 = \texttt{10110101}$

- $\mathcal{S}_0 = 1$
- $\mathcal{S}_1 = 2$
- $\mathcal{S}_2 = 3$
- $\mathcal{S}_3 = 4$
- $\mathcal{S}_4 = 5$
- $\mathcal{S}_5 = 4$

As stated, use induction. The base case is trivial.
Let N_k represent the Fibonacci numbers, with $N_0 = 0$, $N_1 = 1$, and $N_k = N_{k-1} + N_{k-2}$

Assume that F_k has length N_{k+2} for all $k \leq n$. We want to show that F_{k+1} has length N_{k+3}.
Since $F_k = F_{k-1}F_{k-2}$, it has the length $|F_{k-1}| + |F_{k-2}|$.
By our assumption, $|F_{k-1}| = N_{k+1}$ and $|F_{k-2}| = N_k$.
So, $|F_k| = |F_{k-1}| + |F_{k-2}| = N_{k+1} + N_k = N_{k+2}$.

Problem 16:

Let C_k denote the word over the alphabet $\{\texttt{0}, \texttt{1}\}$ obtained by concatenating the binary representations of the integers $0, \ \ldots, \ 2^k - 1$.

For example, $C_1 = \texttt{01}$, $C_2 = \texttt{011011}$, and $C_3 = \texttt{011011100101110111}$.

- Compute $\mathcal{S}_0$, $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$ for C_3.
- Show that $\mathcal{S}_k(C_k) = 2^k - 1$.
- Show that $\mathcal{S}_n(C_k) = 2^n$ for $n < k$.

Hint: If v is a subword of w and w is a subword of u, v must be a subword of u.
In other words, the "subword" relation is transitive.

Solution

$\mathcal{S}_0 = 1$, $\mathcal{S}_1 = 2$, $\mathcal{S}_2 = 4$, and $\mathcal{S}_3 = 7$.

First, we show that $\mathcal{S}_k(C_k) = 2^k - 1$.
Consider an arbitrary word w of length k. We'll consider three cases:

- If w consists only of zeros, w does not appear in C_k.
- If w starts with a $\texttt{1}$, w must appear in C_k by construction.
- If w does starts with a $\texttt{0}$ and contains a $\texttt{1}$, w has the form $\texttt{0}^x\texttt{1}\overline{\texttt{y}}$
  That is, x copies of $\texttt{0}$ followed by a $\texttt{1}$, followed by
  an arbitrary sequence $\overline{\texttt{y}}$ with length $(k - x - 1)$.
  Now consider the word $\texttt{1}\overline{\texttt{y}}\texttt{0}^x\texttt{1}\overline{\texttt{y}}\texttt{0}^{(x-1)}\texttt{1}$.
  This is the concatenation of two consecutive binary numbers with k digits, and thus appears in C_k. w is a subword of this word, and therefore also appears in C_k.

We can use the above result to conclude that $\mathcal{S}_n(C_k) = 2^n$ for $n < k$:
If we take any word of length $n < k$ and repeatedly append $\texttt{1}$ to create a word of length k, we end up with a subword of C_k by the reasoning above.
Thus, any word of length n is a subword of w, of which there are 2^n.

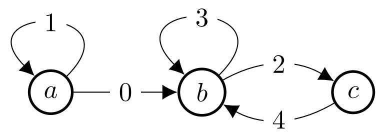
Problem 17:

Convince yourself that C_{n+1} provides a solution to the n-subword problem over $\{\texttt{0}, \texttt{1}\}$.
Note: C_{n+1} may or may not be an *optimal* solution—but it is a *valid* solution
Which part of Problem 16 shows that this is true?

Part 3: De Bruijn Words

Before we continue, we'll need to review some basic graph theory.

Definition 18:
A *directed graph* consists of nodes and directed edges.
An example is shown below. It consists of three vertices (labeled a, b, c),
and five edges (labeled $0, ..., 4$).

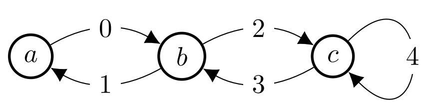


Definition 19:
A *path* in a graph is a sequence of adjacent edges,
In a directed graph, edges a and b are adjacent if a ends at the node which b starts at.

For example, consider the graph above.
The edges 1 and 0 are adjacent, since you can take edge 0 after taking edge 1.
0 starts where 1 ends.
0 and 1, however, are not: 1 does not start at the edge at which 0 ends.

Definition 20:
An *Eulerian path* is a path that visits each edge of a graph exactly once.
An *Eulerian cycle* is an Eulerian path that starts and ends on the same node.

Problem 21:
Find the single unique Eulerian cycle in the graph below.



Solution

24310 is one way to write this cycle.
There are other options, but they're all the same.

Theorem 22:
A directed graph contains an Eulerian cycle iff...
- There is a path between every pair of nodes, and
- every node has as many "in" edges as it has "out" edges.

If the a graph contains an Eulerian cycle, it must contain an Eulerian path. (why?)
Some graphs contain an Eulerian path, but not a cycle. In this case, both conditions above must still hold, but the following exceptions are allowed:
- There may be at most one node where (number in − number out) $= 1$
- There may be at most one node where (number in − number out) $= -1$

Note: Either both exceptions occur, or neither occurs. Bonus problem: why?
We won't provide a proof of this theorem today. However, you should convince yourself that it is true: if any of these conditions are violated, why do we know that an Eulerian cycle (or path) cannot exist?

Definition 23:
Now, consider the n-subword problem over $\{\texttt{0}, \texttt{1}\}$.
We'll call the optimal solution to this problem a *De Bruijn*[1] *word* of order n.

Problem 24:
Let w be the an order-n De Bruijn word, and denote its length with $|w|$.
Show that the following bounds always hold:

- $|w| \leq n2^n$
- $|w| \geq 2^n + n - 1$

Solution

- There are 2^n binary words with length n.
  Concatenate these to get a word with length $n2^n$.
- A word must have at least $2^n + n - 1$ letters to have 2^n subwords with length n.

Remark 25:
Now, we'd like to show that the length of a De Bruijn word is always $2^n + n - 1$
That is, that the optimal solution to the subword problem always has $2^n + n - 1$ letters.
We'll do this by construction: for a given n, we want to build a word with length $2^n + n - 1$ that solves the binary n-subword problem.

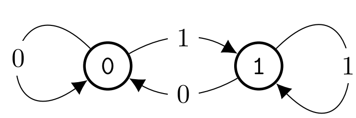
Definition 26:
Consider a n-length word w.
The *prefix* of w is the word formed by the first $n-1$ letters of w.
The *suffix* of w is the word formed by the last $n-1$ letters of w.
For example, the prefix of the word `1101` is `110`, and its suffix is `101`. The prefix and suffix of any one-letter word are both $\varnothing$.

Definition 27:
A *De Bruijn graph* of order n, denoted G_n, is constructed as follows:

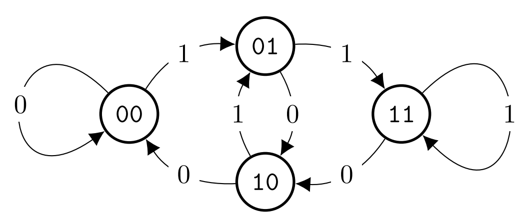
- Nodes are created for each word of length $n-1$.
- A directed edge is drawn from a to b if the suffix of a matches the prefix of b.
  Note that a node may have an edge to itself.
- We label each edge with the last letter of b.

G_2 and G_3 are shown below.

G_2



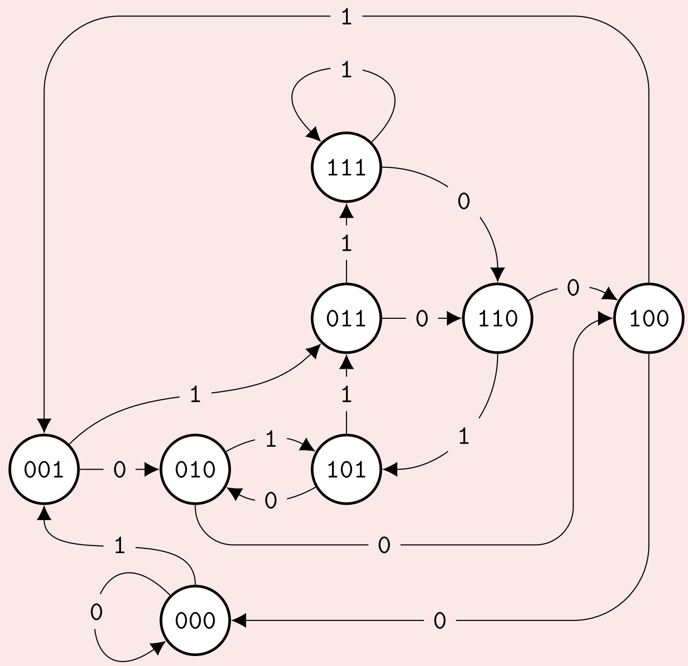
G_3



[1] Dutch. Rhymes with "De Grown."

Problem 28:
Draw G_4.

Solution



Note for Instructors

This graph also appears as a solution to a different problem in the DFA handout.

Problem 29:

- Show that G_n has 2^{n-1} nodes and 2^n edges;
- that each node has two outgoing edges;
- and that there are as many edges labeled 0 as are labeled 1.

Solution

- There 2^{n-1} binary words of length $n - 1$.
- The suffix of a given word is the prefix of two other words,
  so there are two edges leaving each node.
- One of those words will end with one, and the other will end with zero.
- Our 2^{n-1} nodes each have 2 outgoing edges—we thus have 2^n edges in total.

Problem 30:

Show that G_4 always contains an Eulerian path.

Hint: Theorem 22

Theorem 31:

We can now easily construct De Bruijn words for a given n:

- Construct G_n,
- find an Eulerian cycle in G_n,
- then, construct a De Bruijn word by writing the label of our starting vertex, then appending the label of every edge we travel.

Problem 32:

Find De Bruijn words of orders 2, 3, and 4.

Solution

- One Eulerian cycle in G_2 starts at node `0`, and takes the edges labeled $[1, 1, 0, 0]$.
  We thus have the word `01100`.
- In G_3, we have an Eulerian cycle that visits nodes in the following order:
  `00` $\to$ `01` $\to$ `11` $\to$ `11` $\to$ `10` $\to$ `01` $\to$ `10` $\to$ `00` $\to$ `00`
  This gives us the word `0011101000`
- Similarly, we G_4 gives us the word `0001 0011 0101 1110 000`.
  Spaces have been added for convenience.

Let's quickly show that the process described in Theorem 31 indeed produces a valid De Bruijn word.

Problem 33:
How long will a word generated by the above process be?

Solution

A De Bruijn graph has 2^n edges, each of which is traversed exactly once. The starting node consists of $n-1$ letters.

Thus, the resulting word contains $2^n + n - 1$ symbols.

Problem 34:
Show that a word generated by the process in Theorem 31 contains every possible length-n subword. In other words, show that $\mathcal{S}_n(w) = 2^n$ for a generated word w.

Solution

Any length-n subword of w is the concatenation of a vertex label and an edge label. By construction, the next length-n subword is the concatenation of the next vertex and edge in the Eulerian cycle.

This cycle traverses each edge exactly once, so each length-n subword is distinct.
Since w has length $2^n + n - 1$, there are 2^n total subwords.
These are all different, so $\mathcal{S}_n \geq 2^n$.
However, $\mathcal{S}_n \leq 2^n$ by Problem 13, so $\mathcal{S}_n = 2^n$.

Remark 35:

- We found that Theorem 31 generates a word with length $2^n + n - 1$ in Problem 33,
- and we showed that this word always solves the n-subword problem in Problem 34.
- From Problem 24, we know that any solution to the binary n-subword problem must have at least $2^n + n - 1$ letters.
- Finally, Problem 30 guarantees that it is possible to generate such a word in any G_n.

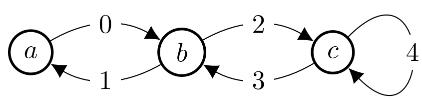
Thus, we have shown that the process in Theorem 31 generates ideal solutions to the n-subword problem, and that such solutions always exist. We can now conclude that for any n, the binary n-subword problem may be solved with a word of length $2^n + n - 1$.

Part 4: Line Graphs

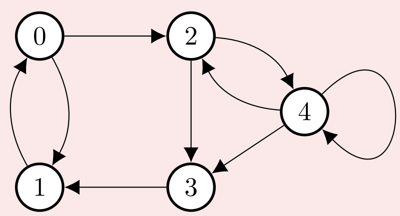
Problem 36:
Given a graph G, we can construct a graph called the *line graph* of G (denoted $\mathcal{L}(G)$) by doing the following:

- Creating a node in $\mathcal{L}(G)$ for each edge in G
- Drawing a directed edge between every pair of nodes a, b in $\mathcal{L}(G)$ if the corresponding edges in G are adjacent.
  That is, if edge b in G starts at the node at which a ends.

Problem 37:
Draw the line graph for the graph below.
Have an instructor check your solution.



Solution



Definition 38:
We say a graph G is *connected* if there is a path between any two vertices of G.

Problem 39:
Show that if G is connected, $\mathcal{L}(G)$ is connected.

Solution

Let a, b and x, y be nodes in a connected graph G so that an edges $a \to b$ and and $x \to y$ exist. Since G is connected, we can find a path from b to x. The path a to y corresponds to a path in $\mathcal{L}(G)$ between $a \to b$ and $x \to y$.

Definition 40:
Consider $\mathcal{L}(G_n)$, where G_n is the n^{th} order De Bruijn graph.

We'll need to label the vertices of $\mathcal{L}(G_n)$. To do this, do the following:
- Let a and b be nodes in G_n
- Let $\mathtt{x}$ be the first letter of a
- Let $\mathtt{y}$, the last letter of b
- Let $\overline{\mathtt{p}}$ be the prefix/suffix that a and b share.
  Note that $a = \mathtt{x}\overline{\mathtt{p}}$ and $b = \overline{\mathtt{p}}\mathtt{y}$,

Now, relabel the edge from a to b as $\mathtt{x}\overline{\mathtt{p}}\mathtt{y}$.
Use these new labels to name nodes in $\mathcal{L}(G_n)$.

Problem 41:
Construct $\mathcal{L}(G_2)$ and $\mathcal{L}(G_3)$. What do you notice?
Hint: What are $\mathcal{L}(G_2)$ and $\mathcal{L}(G_3)$? We've seen them before!
You may need to re-label a few edges.

Solution

After fixing edge labels, we find that $\mathcal{L}(G_2) \cong G_3$ and $\mathcal{L}(G_3) \cong G_4$

Part 5: Sturmian Words

A De Bruijn word is the shortest word that contains all subwords of a given length.
Let's now solve a similar problem: given an alphabet, we want to construct a word that contains exactly m distinct subwords of length n.

In general, this is a difficult problem. We'll restrict ourselves to a special case:
We'd like to find a word that contains exactly $m+1$ distinct subwords of length m for all $m < n$.

Definition 42:
We say a word w is a *Sturmian word* of order n if $\mathcal{S}_m(w) = m+1$ for all $m \leq n$.
We say w is a *minimal* Sturmian word if there is no shorter Sturmian word of that order.

Problem 43:
Show that the length of a Sturmian word of order n is at least $2n$.

Solution

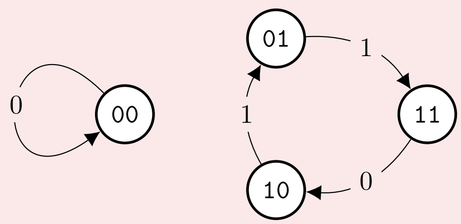
In order to have $n+1$ subwords of length n, a word must have at least $(n+1)+(n-1) = 2n$ letters.

Problem 44:
Construct R_3 by removing four edges from G_3.
Show that each of the following is possible:

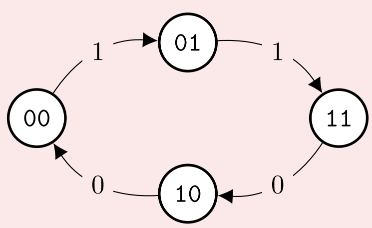
- R_3 does not contain an Eulerian path.
- R_3 contains an Eulerian path, and this path constructs a word w with $\mathcal{S}_3(w) = 4$ and $\mathcal{S}_2(w) = 4$.
- R_3 contains an Eulerian path, and this path constructs a word w that is a minimal Sturmian word of order 3.

Solution

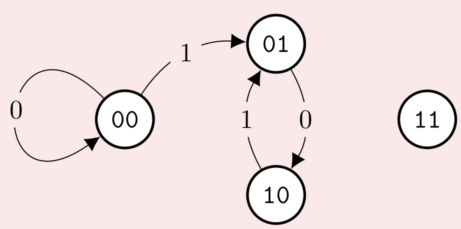
Remove the edges `00` $\to$ `01`, `01` $\to$ `10`, `10` $\to$ `00`, and `11` $\to$ `11`:



Remove the edges `00` $\to$ `00`, `01` $\to$ `10`, `10` $\to$ `01`, and `11` $\to$ `11`.
The Eulerian path starting at `00` produces `001100`, where $\mathcal{S}_2 = \mathcal{S}_3 = 4$.



Remove the edges `01` $\to$ `11`, `10` $\to$ `00`, `11` $\to$ `10`, and `11` $\to$ `11`.
The Eulerian path starting at `00` produces `000101`, where $\mathcal{S}_0 = 1$, $\mathcal{S}_1 = 2$, $\mathcal{S}_2 = 3$, and $\mathcal{S}_3 = 4$.
`000101` has length $2 \times 3 = 6$, and is thus minimal.



Note that this graph contains an Eulerian path even though `11` is disconnected.
An Eulerian path needs to visit all *edges*, not all *nodes*!

Problem 45:

Construct R_2 by removing one edge from G_2, then construct $\mathcal{L}(R_2)$.

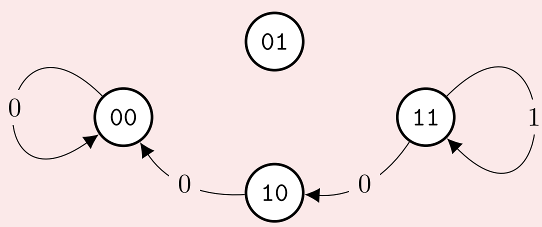
- If this line graph has four edges, set $R_3 = \mathcal{L}(R_2)$.
- If not, remove one edge from $\mathcal{L}(R_2)$ so that an Eulerian path still exists and set R_3 to the resulting graph.

Label each edge in R_3 with the last letter of its target node.
Let w be the word generated by an Eulerian path in this graph, as before.

Attempt the above construction a few times. Is w a minimal Sturmian word?

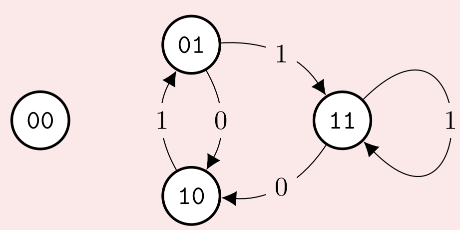
Solution

If R_2 is constructed by removing the edge `0` $\to$ `1`, $\mathcal{L}(R_2)$ is the graph shown below.



We obtain the Sturmian word `111000` via the Eulerian path through the nodes
`11` $\to$ `11` $\to$ `10` $\to$ `00` $\to$ `00`.

If R_2 is constructed by removing the edge `0` $\to$ `0`, $\mathcal{L}(R_2)$ is the graph pictured below.



This graph contains five edges, we need to remove one.
To keep an Eulerian path, we can remove any of the following:

- `10` $\to$ `01` to produce `011101`
- `01` $\to$ `11` to produce `111010`
- `11` $\to$ `10` to produce `010111`
- `11` $\to$ `11` to produce `011010`

Each of these is a minimal Sturmian word.

The case in which we remove `1` $\to$ `0` in G_2 should produce a minimal Sturmian word where `0` and `1` are interchanged in the word produced by removing `0` $\to$ `1`.

If we remove `1` $\to$ `1` will produce minimal Sturmian words where `0` and `1` are interchanged from the words produced by removing `0` $\to$ `0`.

Theorem 46:
We can construct a minimal Sturmian word of order $n \geq 3$ as follows:

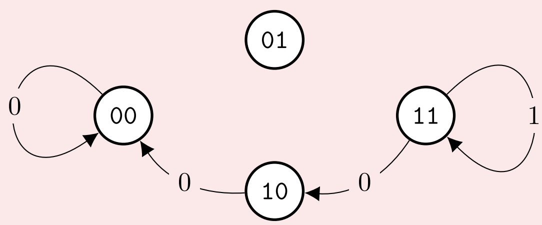
- Start with G_2, create R_2 by removing one edge.
- Construct $\mathcal{L}(G_2)$, remove an edge if necessary.
  The resulting graph must have an 4 edges and an Eulerian path. Call this R_3.
- Repeat the previous step to construct a sequence of graphs R_n.
  R_{n-1} is used to create R_n, which has $n+1$ edges and an Eulerian path.
  Label edges with the last letter of their target vertex.
- Construct a word w using the Eulerian path, as before.
  This is a minimal Sturmian word.

For now, assume this theorem holds. We'll prove it in the next few problems.

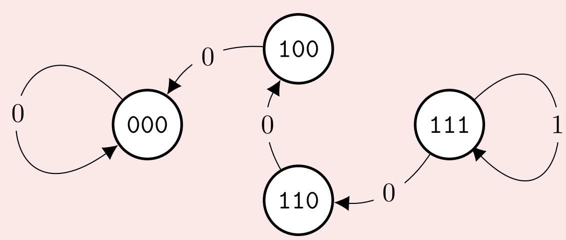
Problem 47:
Construct a minimal Sturmain word of order 4.

Solution

Let R_3 be the graph below (see Problem 45).



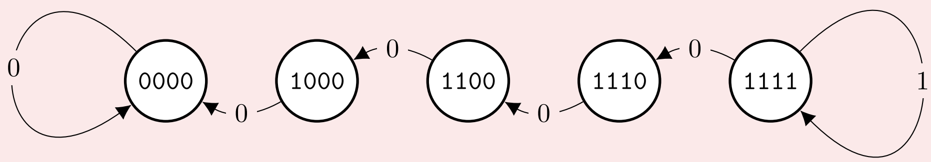
$R_4 = \mathcal{L}(R_3)$ is then as shown below, producing the order 4 minimal Sturman word `11110000`. Disconnected nodes are omitted.



Problem 48:
Construct a minimal Sturmain word of order 5.

Solution

Use R_4 from Problem 47 to construct R_5, shown below.
Disconnected nodes are omitted.



This graph generates the minimal Sturmian word `1111100000`

Problem 49:
Argue that the words we get by Theorem 46 are minimal Sturmain words.
That is, the word w has length $2n$ and $\mathcal{S}_m(w) = m + 1$ for all $m \leq n$.

Solution

We proceed by induction.
First, show that we can produce a minimal order 3 Sturmian word:

R_3 is guaranteed to have four edges with length-2 node labels, the length of w is $2 \times 3 = 6$.
Trivially, we also have $\mathcal{S}_0 = 1$ and $\mathcal{S}_1 = 2$.

There are three vertices of R_3 given by the three remaining nodes of R_2. Each length-2 subword of w will be represented by the label of one of these three nodes. Thus, $\mathcal{S}_2(w) \leq 3$. The line graph of a connected graph is connected, so an Eulerian path on R_3 reaches every node. We thus have that $\mathcal{S}_2(w) = 3$.

By construction, the length 3 subwords of w are all distinct, so $\mathcal{S}_3(w) = 4$. We thus conclude that w is a minimal order 3 Sturmain word.

Now, we prove our inductive step:
Assume that the process above produces an order $n - 1$ minimal Sturmain word w_{n-1}.
We want to show that w_n is also a minimal Sturmain word.

By construction, R_n has node labels of length $n - 1$ and $n + 1$ edges.
Thus, w_n has length $2n$.

The only possilble length-m subwords of w_n are those of w_{n-1} for $m < n$.
The line graph of a connected graph is connected, so an Eulerian path on R_3 reaches each node. Thus, all length-m subwords of w_{n-1} appear in w_n.

By our inductive hypothesis, $\mathcal{S}_m(w_n) = m + 1$ for $m < n$.
The length-n subwords of w_n are distinct by construction, and there are $n + 1$ such subwords.

Thus, $\mathcal{S}_n(w_n) = n + 1$.