ELSEVIER

Discrete Mathematics 226 (2001) 313–336

DISCRETE MATHEMATICS

www.elsevier.com/locate/disc

De Bruijn sequences with efficient decoding algorithms

Jonathan Tuliani[1]

Department of Mathematics, Royal Holloway and Bedford New College, University of London, Egham, Surrey, TW20 0EX, UK

Received 29 June 1998; revised 4 January 2000; accepted 31 January 2000

Abstract

A de Bruijn sequence of span n is a periodic sequence in which every n-tuple over the alphabet appears exactly once, starting within the minimal generating cycle of the sequence. The decoding problem for such a sequence is to determine the position of an arbitrary n-tuple over the alphabet. Algorithms that allow this problem to be solved efficiently are important in certain types of position sensing applications. We present constructions for de Bruijn sequences over a variety of alphabets, together with corresponding decoding algorithms, that constitute a significant improvement over existing methods. These results include a construction for binary de Bruijn sequences of arbitrary span whose decoding algorithm is both fast and storage-free.

Keywords: de Bruijn sequence; Construction; Decoding; Position sensing

1. Introduction

A de Bruijn sequence of span n is a periodic sequence in which every possible n-tuple over the alphabet occurs exactly once as a window starting within the minimal generating cycle of the sequence. These sequences have been proposed for use in certain position sensing schemes, in which a moving device reads an n-window from a fixed representation of the minimal generating cycle of the sequence. The uniqueness of these windows allows the device to determine its position. Implicit in this is the assumption that the device has a means of calculating the position within the sequence of the n-window that it reads. This problem is known as the decoding problem for the sequence. The use of sequences with efficient decoding algorithms is important in such schemes, as long decoding times may limit the speed with which the device may be

[1] This research was completed as part of a Ph.D. Degree supported by a CASE studentship from the Engineering and Physical Sciences Research Council and Hewlett-Packard Laboratories, Bristol.
E-mail address: jonathan.tuliani@hushmail.com (J. Tuliani).

0012-365X/01/$ - see front matter
PII: S0012-365X(00)00117-5

allowed to move [6]. De Bruijn sequences are also interesting mathematical structures in their own right, and, as noted in [4], the decoding problem for de Bruijn sequences has been listed as one of the 'fundamental questions' for their further study [1].

Until recently, all known techniques for decoding de Bruijn sequences had complexity or storage requirements that are non-polynomial (and usually exponential) in the span n. A summary of these techniques is given in [4], whose authors go on to exhibit the first decoding algorithm for de Bruijn sequences whose complexity and storage requirements are polynomial (and of low order) in n. Whilst theoretically significant, their result is of limited practical value as it is applicable for only a small subset of the possible values of n.

Section 3 gives a construction for de Bruijn sequences of arbitrary span over alphabets of arbitrary size, together with an associated decoding procedure. This work is based on previous results in the binary case [2,5]. The construction of [4] is introduced in Section 4, and in Section 5 is combined with our earlier technique to give a hybrid construction that has distinct advantages over each of its precursors. In Section 6 this hybrid construction is then analysed more closely in the binary case. It is shown that the storage requirements of the decoding algorithm can be eliminated, at the expense of a small extra amount of computation. The result is a construction for binary de Bruijn sequences of arbitrary span n, whose decoding algorithm has a complexity of $O(n \log n)$ integer operations, and which requires just $O(1)$ bits of storage (Theorem 17). Section 7 gives a means of constructing de Bruijn sequences of arbitrary span n over alphabets of arbitrary composite size c, such that the decoding problem for these sequences may be reduced to the problem of decoding a span n de Bruijn sequence over an alphabet of size d, where d is any non-trivial factor of c (Theorem 24). We conclude by combining the results of Sections 6 and 7, showing how we may construct de Bruijn sequences of arbitrary span over alphabets of arbitrary even size that may be decoded quickly and without the use of storage tables. Finally, we consider what open problems remain in this area.

2. Preliminaries

A sequence $[s]$ is a bi-infinite list $\ldots, s_{-1}, s_0, s_1, \ldots$ of elements taken from a set $\mathscr{A}$. We shall be concerned with periodic sequences, namely sequences for which there exists a positive integer ℓ such that $s_i = s_{i+\ell}$ for all $i \in \mathbb{Z}$. Any such sequence may be defined by specifying any ℓ consecutive terms. By convention we start with the term indexed by zero, writing $[s] = [s_0, s_1, \ldots, s_{\ell-1}]$. We call $[s_0, s_1, \ldots, s_{\ell-1}]$ a *generating cycle* for $[s]$. We also say $[s]$ has *period* ℓ. The least such ℓ is called the *least period* of $[s]$, and the generating cycle of this length is described as being *minimal*.

The set $\mathscr{A}$ we refer to as the *alphabet*. We shall be concerned only with alphabets whose size (cardinality) is both finite and at least 2. Whenever $|\mathscr{A}| = c$ we may say that the sequence $[s]$ is *c-ary*. We will often use the ring $\mathbb{Z}_c$ as an example of a c-ary alphabet. For a sequence $[s]$ of least period ℓ over $\mathbb{Z}_c$ we define the *generalised weight*

of $[s]$, denoted $\mathrm{wt}([s])$, by $\mathrm{wt}([s]) = \sum_{i=0}^{\ell-1} s_i$, the result lying in $\mathbb{Z}_c$. The following lemma is simple to prove.

Lemma 1. *Let n and c be positive integers, with $n \geqslant 1$ and $c \geqslant 2$, and let $[s]$ be a span n de Bruijn sequence over $\mathbb{Z}_c$. Then $\mathrm{wt}([s]) = 0$ if and only if $n \geqslant 2$ or $n = 1$ and $2 \nmid c$.*

For any $v \in \mathscr{A}$, we write $v_{|n|}$ to denote the n-tuple all of whose entries are equal to v. If $x = (x_0, x_1, \ldots, x_{n-1}) \in \mathscr{A}^n$ has the property that, for some $k \in \mathbb{Z}$, we have $x_i = s_{k+i}$ for $i = 0, 1, \ldots, n-1$ then we say that x is an *n-window* of $[s]$, and that x *appears in* $[s]$ *at position* k. Clearly, given $[s]$, n and k there is a unique n-tuple x with this property, which we denote $[s]_k^n$.

Any window appearing in a periodic sequence must do so an infinite number of times. Thus, we are free to say that w is a *unique* window in a sequence $[s]$ of least period ℓ, or that w appears *exactly once* in $[s]$ when we mean, strictly, that w appears at exactly one position k in the range $0 \leqslant k \leqslant \ell - 1$. We write $\mathrm{loc}([s]w)$ to denote the least, positive position of the n-tuple w in $[s]$.

We now define the difference operator D on sequences over $\mathbb{Z}_c$ and present a summary of its properties. These are simple generalisations from the binary case of results that may be found in [2].

Let $[t]$ be any sequence over $\mathbb{Z}_c$. We define the sequence $[s]$ over $\mathbb{Z}_c$ by $s_i = t_{i+1} - t_i$ for all $i \in \mathbb{Z}$, and write $[s] = D[t]$. We extend the operator D to act also as a map from $\mathbb{Z}_c^n$ to $\mathbb{Z}_c^{n-1}$, for any $n \geqslant 2$. Given $v = (v_0, v-1, \ldots, v_{n-1}) \in \mathbb{Z}_c^n$ and $u = (u_0, u_1, \ldots, u_{n-2}) \in \mathbb{Z}_c^{n-1}$, we write $u = Dv$ in the event that $u_i = v_{i+1} - v_i$ for $i = 0, 1, \ldots, n-2$. Clearly, for any $n \geqslant 2$, the n-tuple v appears in the sequence $[t]$ at position $p \in \mathbb{Z}$ only if Dv appears in $D[t]$ at position p.

Let $u = (u_0, u_1, \ldots, u_{n-2})$ be any element of $\mathbb{Z}_c^{n-1}$, and let $[s]$ be a sequence over $\mathbb{Z}_c$ with least period ℓ and generalised weight w. The c pre-images of u and $[s]$ under D are given by $v = (v_0, v_1, \ldots, v_{n-1}) \in \mathbb{Z}_c^n$ and $[t]$, respectively, where

$$v_i = \begin{cases} b & \text{if } i = 0, \text{ or} \\ b + \displaystyle\sum_{j=0}^{i-1} u_j & \text{if } 1 \leqslant i \leqslant n-1, \end{cases}$$

$$t_i = \begin{cases} b & \text{if } i = 0, \\ b + \displaystyle\sum_{j=0}^{i-1} s_j & \text{if } i > 0, \text{ or} \\ b - \displaystyle\sum_{j=i}^{-1} s_j & \text{if } i < 0, \end{cases}$$

and where b is any element of $\mathbb{Z}_c$. Moreover, the least period of $[t]$ is ℓ if $w = 0$ and $c\ell/\gcd(c, w)$ otherwise. We write D_b^{-1} to denote the map which returns the pre-image identified by the given value b.

We will be particularly concerned with pre-images of c-ary sequences $[s]$ of generalised weight w and least period ℓ for which $\gcd(c,w)=1$. In this case, each pre-image is a shift of $D_0^{-1}[s]$. For any $(n-1)$-tuple u appearing exactly once in $[s]$, the c pre-images of u under D each appear in $D_0^{-1}[s]$. Moreover they are equally spaced in $D_0^{-1}[s]$ and their order is determined by w — explicitly, if $D_b^{-1}u$ appears in $D_0^{-1}[s]$ at position h then $D_{b+w}^{-1}u$ appears in $D_0^{-1}[s]$ at position $h+\ell$, $D_{b+2w}^{-1}u$ appears in $D_0^{-1}[s]$ at position $h+2\ell$, and so on.

3. The generalised Lempel construction

In [2], Lempel presents a construction for binary de Bruijn sequences based on the operator D. Paterson and Robshaw have shown how these sequences may be decoded [5]. This section generalises both techniques to alphabets of arbitrary size.

Let c,k and ℓ be positive integers, with $c,\ell\geqslant 2$ and $0\leqslant k\leqslant \ell-1$, and let $[s]$ be a sequence over $\mathbb{Z}_c$ of least period ℓ. We define the map χ_k on $[s]$ to return the sequence of period $\ell-1$ given by the generating cycle $[s_0,s_1,\ldots,s_{k-1},s_{k+1},s_{k+2},\ldots,s_{\ell-1}]$. For $[s]$, c, and ℓ as above, and for any integer p in the range $0\leqslant p\leqslant \ell$ and any $e\in\mathbb{Z}_c$, we also define the map $\rho_{p,e}$ on $[s]$ to return the sequence of period $\ell+c$ over $\mathbb{Z}_c$ given by the generating cycle $[s_0,s_1,\ldots,s_{p-1},e,e+1,\ldots,e+c-1,s_p,s_{p+1},\ldots,s_{\ell-1}]$ (where the expressions in e are, of course, reduced modulo c).

Construction 2. *Let n and c be integers, with $n\geqslant 1$ and $c\geqslant 2$ and with $2\nmid c$ if $n=1$. Let $[s]$ be a span n de Bruijn sequence over $\mathbb{Z}_c$. We define the sequence $[t]$ as follows*:

(1) *let* $k=\mathrm{loc}([s],1_{|n|})$;
(2) *let* $[s']=\chi_k[s]$;
(3) *let* $[\hat{s}]=D_0^{-1}[s']$;
(4) *let* $p=(c-1)(c^n-1)+k$ *and let* $e=\hat{s}_p$, *and*
(5) *let* $[t]=\rho_{p,e}[\hat{s}]$.

Remark 3. Since $[s]$ is a de Bruijn sequence we have that the least period of $[s']$ is c^n-1. By Lemma 1, and since $s_k=1$, we have that $\mathrm{wt}([s'])=c-1$, and since $\gcd(c,c-1)=1$ we have that $[\hat{s}]$ has least period $c(c^n-1)$. Since $k\leqslant c^n-1$ we have $p\leqslant c\,(c^n-1)$, and so step 5 of Construction 2 is well defined.

Lemma 4. *The sequence $[t]$ given by Construction 2 is a de Bruijn sequence of span $n+1$ over $\mathbb{Z}_c$.*

Proof. Every n-tuple except for $1_{|n|}$ appears as an n-window in $[s']$, and every pre-image of these n-tuples under D appears as an $(n+1)$-window of $[\hat{s}]$. Thus the only $(n+1)$-tuples not appearing as windows of $[\hat{s}]$ are the $(n+1)$-tuples $x_b=D_b^{-1}(1_{|n|})$ for each $b\in\mathbb{Z}_c$.

Since $1_{|n-1|}$ appears in $[s']$ at position k, we must have the n-tuple $(v, v+1, \ldots, v+n-1)$ (terms modulo c) appearing at position k in $[\hat{s}]$, where $v = \hat{s}_k$. Since $[s']$ has least period $c^n - 1$, a similar n-tuple $(e, e+1, \ldots, e+n-1)$ (terms modulo c) appears at position $p = (c-1)(c^n-1)+k$ in $[\hat{s}]$, where $e = \hat{s}_p$. We now consider what happens to $[\hat{s}]$ when we apply the map $\rho_{p,e}$ to obtain the sequence $[t]$. Clearly, for $0 \leqslant j \leqslant p-n-1$, we have $[\hat{s}]_j^{n+1} = [t]_j^{n+1}$. By choosing p and e so that x_e appears at position p, and by the definition of $\rho_{p,e}$, we have that $\hat{s}_\ell = t_\ell$ for $p \leqslant \ell \leqslant p+n-1$. (It may be useful to consider the cases $n < c$, $n = c$ and $n > c$ separately.) Thus $[\hat{s}]_j^{n+1} = [t]_j^{n+1}$ for $p-n \leqslant j \leqslant p-1$. Also, $[t]_j^{n+1} = x_{e+j-p \bmod c}$ for $p \leqslant j \leqslant p+c-1$. Finally, since $[t]$ has the same generating cycle as $[\hat{s}]$ except for the insertion of a c-tuple at position p, we have that $[\hat{s}]_j^{n+1} = [t]_{j+c}^{n+1}$ for $p \leqslant j \leqslant c^{n+1} - c - 1$. Notice that every $(n+1)$-window of $[\hat{s}]$ appears exactly once in $[t]$, as does each $(n+1)$-tuple x_b. Thus $[t]$ is a span $n+1$ de Bruijn sequence as required. □

Example 5. Let $[s]$ be the span 3 de Bruijn sequence over $\mathbb{Z}_3$ given by

$$[s] = [000222122021121020120011101].$$

Notice that $k = \mathrm{loc}([s], 1_{|3|}) = 22$, and so $[s'] = \chi_k[s]$ is given by

$$[s'] = [000222122021121020120011101].$$

Then $[\hat{s}] = D_0^{-1}[s']$ is

$$\begin{aligned}[\hat{s}] = [&00002101022120200220222011\\ &22221020211012122112111200\\ &11110212100201011001000\underline{1}22].\end{aligned} \tag{1}$$

We have $p = 2 \times 26 + 22 = 74$, and so $e = \hat{s}_p = 0$ (this digit is underlined in Eq. (1) above). Inserting the 3-tuple (012) at position 74 gives

$$\begin{aligned}[t] = [&00002101022120200220222011\\ &22221020211012122112111200\\ &11110212100201011001000120122],\end{aligned}$$

a de Bruijn sequence of span 4 over $\mathbb{Z}_3$.

We write $[t] = \Phi[s]$ in the event that $[t]$ is a span $n+1$ de Bruijn sequence obtained from a span n de Bruijn sequence $[s]$ using Construction 2. In the case $c = 2$ the sequences that result are similar to those arising from Lempel's original construction, and can be made identical if the value of p is chosen slightly differently.

As with Lempel's technique, the procedure described above for constructing a span $n+1$ de Bruijn sequence from one of span n may be applied repeatedly. That is, let $c \geqslant 2$ be a fixed alphabet size, and let r and n be integers, with $n > r \geqslant 1$ and $2 \nmid c$ if $r = 1$. Then let $[s^{(0)}]$ be a span r de Bruijn sequence over $\mathbb{Z}_c$. We define the sequences $[s^{(i+1)}]$ for $i = 0, 1, \ldots, n-r$ by $[s^{(i+1)}] = \Phi[s^{(i)}]$. (We also define the sequences $[s(i)']$

and $[\hat{s}^{(i)}]$ to be those intermediate sequences arising in the construction of $[s^{(i+1)}]$ from $[s^{(i)}]$, and define $k^{(i)} = \mathrm{loc}([s^{(i)}], 1_{|r+i|}.)$ Then $[s^{(n-r)}]$ is a de Bruijn sequence of span n over $\mathbb{Z}_c$. We now present a procedure for decoding this sequence, based on Paterson and Robshaw's procedure for decoding Lempel's original binary de Bruijn sequences [5].

First, we impose the extra condition that the initial sequence used above should have a window of r zeroes at position 0, that is $[s^{(0)}]_0^r = 0_{|r|}$. By our construction method, this ensures that $[s^{(i)}]_0^{r+i} = 0_{|r+i|}$ for all $i \geqslant 0$. Keeping track of the position of the longest run of zeroes in each sequence allows us to keep track of its pre-images in the next sequences, namely the maximal runs of a constant digit. In particular, since $[s^{(i)\prime}]$ has generalised weight $c-1$ and least period $c^{r+i}-1$, we have $[\hat{s}^{(i)}]_{(c-1)(c^{r+i}-1)}^{r+i+1} = 1_{|r+i+1|}$ for $i = 1, 2, \ldots, n-r$. Now since $k^{(i)} \geqslant r+i$, we have $(c-1)(c^{r+i}-1) + r + i \leqslant (c-1)(c^{r+i}-1) + k^{(i)}$. Thus the formation of $[s^{(i+1)}]$ by step 5 of Construction 2 does not affect the digits of $[\hat{s}^{(i)}]_{(c-1)(c^{r+1}-1)}^{r+i}$, and we have that $[s^{(i+1)}]_{(c-1)(c^{r+i}-1)}^{r+i+1} = 1_{|r+i+1|}$. Thus

$$k^{(i)} = (c-1)(c^{r+i-1}-1) \tag{2}$$

for all $i = 1, 2, \ldots, n-r$.

Suppose we have stored the following information:

(1) c-ary digits $L^{(i)}(j) = s_j^{(i)}$ for each i and j in the range $1 \leqslant i \leqslant n-r$ and $0 \leqslant j \leqslant c^{i+r-1} - 2$;
(2) integers $T(w) = \mathrm{loc}([s^{(0)}], w)$ for all $w \in \mathbb{Z}_c^r$.

The set of integers $\{T(w)\}$ is simply a table allowing the sequence $[s^{(0)}]$ to be decoded immediately. (If available, this could be replaced by a decoding algorithm for the sequence $[s^{(0)}]$.) We may find $k^{(0)}$ from this table, since $k^{(0)} = T(1_{|r|})$.

For any $i \geqslant 1$, we have that the digits $L^{(i)}(j)$ are simply the first $c^{r+i-1}-1$ digits of $[\hat{s}^{(i-1)}]$. Any element of $[s^{(i)}]$ may be found from $k^{(i-1)}$ and a single value $L^{(i)}(j)$. Explicitly, we have

$$\hat{s}_j^{(i-1)} = L^{(i)}(j \bmod (c^{r+i-1}-1)) + (c-1)\left\lfloor \frac{j}{c^{r+i-1}-1} \right\rfloor \quad \text{for all } j \in \mathbb{Z}, \tag{3}$$

and given $k^{(i-1)}$ (to determine the value p used in step 5 of Construction 2), it is then simple to compute the terms of $[s^{(i)}]$ from those of $[\hat{s}^{(i-1)}]$.

Algorithm 1. *Given the stored information described above, it is possible to implement a recursive decoding algorithm for* $[s^{(n-r)}]$. *Suppose we wish to compute* $j = \mathrm{loc}([s^{(n-r)}], w)$ *for some* $w = (w_0, w_1, \ldots, w_{n-1}) \in \mathbb{Z}_c^n$. *Firstly, compute* $v = Dw \in \mathbb{Z}_c^{n-1}$. *If* $v = 1_{|n-1|}$ *then* w *is one of the* n*-tuples introduced into* $[s^{(n-r)}]$ *in step* 5 *of Construction* 2. *Given* p (*which may be found from* $k^{(n-r-1)}$) *and* $e = \hat{s}_p^{(n-r-1)}$ (*which may be found using Eq.* (3)) *the position of* w *in* $[s^{(n-r)}]$ *may be computed as* $j = p + ((w_0 - e) \bmod c)$ (*where the term in parentheses is considered as an integer rather than as an element of* $\mathbb{Z}_c$).

Alternatively, suppose that $v \neq 1_{|n-1|}$. *Then the position of* v *in* $[s^{(n-r-1)}]$ *may be found by applying this algorithm recursively, with the value of* n *reduced by one. The position of* v *in* $[s^{(n-r-1)}]$ *gives rise to its position* f *say in* $[s'^{(n-r-1)}]$. *There are* c *pre-images of* v *under* D, *giving* c *possible positions of* w *in* $[\hat{s}^{(n-r-1)}$. *Since* $\mathrm{wt}([s'^{(n-r-1)}]) = c-1$, *we have* $\mathrm{loc}([\hat{s}^{(n-r-1)}], w) = f + (L^{(n-r)}(j) - w_0)(c^{n-1}-1)$. *Given this value, together with the value of* p (*obtained from* $k^{(n-r-1)}$), *the position of* w *in* $[s^{(n-r)}]$ *is easily found.*

The recursion continues until either the case $v = 1_{|n-1|}$ *above occurs, or until* $n = r$, *at which point* $j = T(w)$.

We may now give our main result of this Section.

Theorem 6. *Let* n *and* c *be positive integers, with* $c \geqslant 2$ *and* $n \geqslant 3$. *Then there exists a de Bruijn sequence of span* n *over* $\mathbb{Z}_c$, *whose decoding procedure has a computational complexity of* $\mathrm{O}(n^2)$ *integer operations, and which requires* $\mathrm{O}(c^n \log c)$ *bits of storage.*

Proof. Excluding the recursion, the algorithm uses no looping procedures. Considering the recursion, we see that each decoding of $[s^{(n-r)}]$ requires at most one decoding of each of the sequences $[s^{(n-r-1)}], [s^{(n-r-2)}], \ldots, [s^{(r+1)}]$, and possibly one look-up in the table T (to perform a decoding of $[s^{(r)}]$). The recursion decoding $[s^{(i-r)}]$ requires $\Theta(i)$ integer operations, the most expensive step being the computation of v from w. Thus the algorithm has an overall complexity of $\mathrm{O}(n(n-r))$ integer operations.

The storage requirements for the decoding algorithm are as follows:

(1) the values $L^{(i)}(j)$, which require the storage of $c^{r+i-1}-1$ c-ary digits for each i in the range $1 \leqslant i \leqslant n-r$;
(2) the values $T(w)$, which require the storage of c^r c^r-ary digits.

Thus the total storage requirement is for $\mathrm{O}((c^n + rc^r)\log c)$ bits. Letting $r = 2$, we obtain the complexity and storage measures given in the theorem. □

Decoding using a naïve look-up table requires c^n c^n-ary digits, and so uses $\mathrm{O}(nc^n \log c)$ bits of storage. The above method reduces this by a factor of n. It is possible to reduce this requirement further, approximately by a factor of c. We omit the details, as they are not relevant to the main direction of this paper; they may be found in [7].

4. The interleaving construction

This section presents a brief description of a construction for de Bruijn sequences due to Mitchell et al. [4], which we shall call the interleaving construction. This technique takes a de Bruijn sequence of span n and, in essence, interleaves two copies of it to obtain a de Bruijn sequence of span $2n$. (Some small changes are made to the sequences involved both before and after interleaving; these are described below.)

Given two sequences $[b]$ and $[a]$, we define the *interleaving* of $[b]$ and $[a]$, denoted $\mathscr{I}([b],[a])$, to be the sequence $[s]$ given by

$$s_i = \begin{cases} b_{i/2} & \text{if } i \text{ is even, or} \\ a_{(i-1)/2} & \text{if } i \text{ is odd.} \end{cases}$$

Let $[t]$ be a span n de Bruijn sequence over $\mathbb{Z}_c$, where n and c are any integers with $n, c \geqslant 2$. Mitchell et al. present two separate interleaving constructions, according as to whether c is even or odd. We present the former, slightly more complex case, as this is the case we shall consider in more detail when considering binary alphabets later. The latter case is similar [4, Parts IIA and IIB]. Let X_r denote the r-tuple $(1,0,1,0\ldots)$, for all $r \geqslant 1$.

Construction 7 (Mitchell et al. [4, Section III]). *Let n and c be integers, with $n, c \geqslant 2$. Suppose that c is even and that $[t]$ is a span n de Bruijn sequence over $\mathbb{Z}_c$, in which $0_{|n|}$ occurs at position 0 and $1_{|n|}$ occurs at position k. We form the sequence $[b]$ of least period $c^n + 2$ by inserting an extra '1' at position k of the minimal generating cycle of $[t]$ and an extra '0' at position 0. We also define the sequence $[a]$ of period $c^n - 2$ by 'doubly puncturing' $[t]$, that is by removing a '1' from position k and a '0' from position 0 of the minimal generating cycle of $[t]$. Next, we form the sequence $[d]$ of least period $(c^n - 2)(c^n + 2) = c^{2n} - 4$ by letting*

$$[d] = \mathscr{I}([b],[a]).$$

We note that the $(2n-1)$-tuples X_{2n-1} and $1_{|2n-1|}$ must each occur exactly once in $[d]$, and denote their respective positions by v and m. Finally, we let $[s]$ be the sequence whose minimal generating cycle is formed by (simultaneously) inserting an extra '10' into the minimal generating cycle of $[d]$ at position v, inserting a '1' at position m and a '0' at position 0. The sequence $[s]$ is a de Bruijn sequence with span $2n$.

We write $[s] = \Upsilon[t]$ to denote that $[s]$ is a span $2n$ de Bruijn sequence formed using interleaving from a span n de Bruijn sequence $[t]$ (by the above method when the alphabet size c is even and by the related method of [4] when c is odd).

Result 8. Mitchell et al. [4, Sections IIE,IIIE]

Let $[t]$ be a span de Bruijn sequence, and let $[s] = \Upsilon[t]$. Then there exists a decoding procedure for $[s]$ that requires:

(1) *at most two decodings of $[t]$;*
(2) *the solution modulo $c^{2n} - \varepsilon$ of a pair of simultaneous congruences, given modulo $c^n - \varepsilon$ and $c^n + \varepsilon$ (where ε represents some small, variable integer), and*
(3) *a small, constant amount of computation and storage.*

Suppose that a de Bruijn sequence of span n is required, where $n = a2^b$ where a is odd. Given an initial sequence $[t]$ of span a, applying the interleaving construction

b times produces a de Bruijn sequence of span n, whose decoding problem may be reduced to 2^b decodings of the sequence $[t]$ using Result 8. Thus when used in isolation the interleaving technique is effective only when a is small. In particular, it is of no use when the required span n is odd.

5. A hybrid construction

We demonstrate how Construction 2 and the interleaving construction may be used in conjunction with each other, to give a hybrid construction that has distinct advantages over each of its precursors.

Construction 9 (The hybrid construction). *We aim to construct a c-ary de Bruijn sequence of span n, where n and c are integers with* $n, c \geqslant 2$.

Firstly, let $[s^{(2)}]$ *be any span* 2 *de Bruijn sequence over* $\mathbb{Z}_c$, *with* $[s^{(2)}]_0^2 = 0_{|2|}$. *Let* $d = \lfloor 1 + \log_2 n \rfloor$ *be the number of digits in the binary representation of n, and denote this representation by* $n_1 n_2 \cdots n_d$ (*with the most significant digit to the left*). *Necessarily,* $n_1 = 1$ *and* $d \geqslant 2$.

Next, inductively define the sequences $[s^{(i)}]$ *and* $[t^{(i)}]$ for $2 \leqslant i \leqslant d$ *by*

$$[s^{(i)}] = \Upsilon[t^{(i-1)}] \text{ and} \tag{4}$$

$$[t^{(i)}] = \begin{cases} [s^{(i)}] & \text{if } n_i = 0, \text{ or} \\ \Phi[s^{(i)}] & \text{if } n_i = 1. \end{cases} \tag{5}$$

Note that the necessary conditions for the application of Construction 2 are always satisfied, since it is only applied to sequences of span at least 2. Defining the integers m_i by $m_i = \sum_{j=1}^{i} n_j 2^{i-j}$, we have that $[t^{(i)}]$ is a de Bruijn sequence of span m_i for all i in the range $2 \leqslant i \leqslant d$. In particular, $[t^{(d)}]$ has span n.

Each construction used in Construction 9 is a means of forming a higher span sequence from a lower span one, in such a way that the decoding problem for the higher span sequence may easily be solved assuming it is possible to decode the lower span sequence. Where previously these techniques had been used in isolation, we shall exploit the fact that sequences formed by a combination of these methods also allow for recursive decoding, using an appropriate combination of the decoding methods for the separate constructions.

Theorem 10. *Let n and c be arbitrary integers, with* $n, c \geqslant 2$, *and let* $d = \lfloor 1 + \log_2 n \rfloor$. *Let a and b be positive integers such that* $n = a2^b$, *where a is odd, and let* $[t^{(d)}]$ *be the c-ary de Bruijn sequence of span n arising from Construction* 9. *Then the decoding procedure for* $t^{(d)}$ *described above has a computational complexity of* $O(n \log n(1 + \log c))$ *integer operations, and requires* $O((c^a + n)\log c)$ *bits of storage.*

Proof. To decode $[t^{(i)}]$ requires $O(m_i)$ integer operations, together with a decoding of $[s^{(i)}]$. By Result 8, decoding $[s^{(i)}]$ requires two decodings of $[t^{(i-1)}]$, together with the solution of a pair of simultaneous congruences. Since these have a least modulus of size at most c^{2^i-1}, they may be solved using Euclid's Algorithm in at most $O(c^{2^i-1})$ integer operations. Summing these requirements for $2 \leqslant i \leqslant d$ we have that the algorithm requires

$$\begin{aligned} O\left(\sum_{i=2}^{d} [2^{d-i} m_i + 2^{d-i} \log c^{2^i}]\right) &= O\left(\sum_{i=2}^{d} [2^{d-i}(2^i - 1) + 2^d \log c]\right) \\ &= O\left(\sum_{i=2}^{d} [2^d + 2^d \log c]\right) \\ &= O(n \log n(1 + \log c)) \end{aligned}$$

integer operations.

The storage required for each decoding of a sequence formed using the interleaving construction is described in [4, Section IIIE]. This states that, with the notation of Construction 7, it is necessary to store three integers, corresponding to the position of the digits inserted into $[d]$ to form $[t]$. By always having the all-zero n-tuple at position 0 in all our sequences we have already removed the need to store one of these values. The other two values, denoted v and m in Construction 7, must be stored for each of the $O(d)$ applications of the Construction used to form $[t^{(d)}]$. This requires $O(n \log c)$ bits of storage.

The application of a single recursion of Algorithm 1 to decode $[t^{(i)}]$ requires the storage of certain tables of information, namely

(1) c-ary digits $L^{(i)}(j) = t_j^{(i)}$ for each j in the range $0 \leqslant j \leqslant c^{m_i - 1} - 2$, and
(2) a c^{m_i}-ary digit $k^{(i)} = \mathrm{loc}([t^{(i)}], 1_{|m_i|})$.

However, these values need only be stored for the values of i in the range $2 \leqslant i \leqslant d$ for which $n_i = 1$. These storage requirements are dominated by the table $L^{(i)}(j)$ corresponding to the largest value i for which $n_i = 1$, which requires $O(c^a \log c)$ bits of storage. The result follows. □

This asymptotic complexity is less than that given in Theorem 6, and in the worst case, where $a = n$, this storage requirement is asymptotically equal to that given in Theorem 6. Moreover, each additional factor of 2 in n reduces this storage requirement significantly. Thus the advantages of Construction 9 over Construction 2 are clear. Also, Construction 9 may be used for any value n, in contrast to the small subset of possible values that may be used when the interleaving construction is used alone.

6. The binary case

In this section we consider only the binary alphabet. It is our intention to demonstrate how, in this case, given some stored tables of small, constant size the storage

required for decoding sequences formed using Construction 9 may be replaced by some calculations of low computational complexity. To do so, we shall require a number of preliminary results.

Given any sequence $[r]$ we define the sums $\sigma_j[r] \in \mathbb{Z}_2$ for all $j \geqslant 1$ by

$$\sigma_j[r] = \sum_{i=0}^{j-1} r_i.$$

We also define the double sums $\eta_j[r] \in \mathbb{Z}_2$ for all $j \geqslant 2$ by

$$\eta_j[r] = \sum_{i=1}^{j-1} \sigma_i[r].$$

We will find sums of alternate elements from sequences useful, and so define $\xi_e([r], j) \in \mathbb{Z}_2$ and $\xi_o([r], j) \in \mathbb{Z}_2$ for all $j \geqslant 1$ by

$$\xi_e([r], j) = \sum_{i=0}^{j-1} r_{2i},$$

$$\xi_o([r], j) = \sum_{i=0}^{j-1} r_{2i+1}.$$

The subscripts 'e' and 'o' refer to the summations being over the terms whose indices are even or odd, respectively. The role of j in the expressions $\sigma_j[r]$, $\xi_e([r], j)$ and $\xi_o([r], j)$ is to denote the number of terms in each of these summations, thus we define $\sigma_0[r] = \xi_e([r], 0) = \xi_o([r], 0) = 0$. Notice that any expressions for each of the sums σ_j, η_j, ξ_e and ξ_o described above should be taken modulo 2. The following remark is of central importance in what shall follow.

Remark 11. For any sequence $[r]$ and any $j \geqslant 2$ we have that, since our alphabet is binary,

$$\begin{aligned}
\eta_j[r] &= \sum_{i=1}^{j-1} \sigma_i[r] \\
&= \sum_{i=1}^{j-1} \sum_{\ell=0}^{i-1} r_\ell \\
&= \sum_{i=0}^{j-2} (j-1-i) r_i \\
&= \begin{cases} \xi_e([r], j/2) & \text{for } j \text{ even, or} \\ \xi_o([r], (j-1)/2) & \text{for } j \text{ odd.} \end{cases}
\end{aligned}$$

We now give an outline of the strategy we shall adopt for eliminating the storage requirement of Theorem 10. Imagine for the moment that the interleaving construction

consists purely of interleaving a de Bruijn sequence with a copy of itself, and ignore the extra stages involving the insertion and deletion of other digits. Similarly, imagine that Construction 2 simply involves applying the D_0^{-1} operator (that is, forming a new sequence by taking partial sums of terms from the previous sequence), again ignoring the stages involving the insertion and removal of other digits.

Now suppose for each sequence $[s^{(i)}]$ arising in Construction 9 that the terms, sums of terms and double sums of terms, as represented by the notation σ and η established above, are known. For each i, the sequence $[t^{(i)}]$ is formed either by copying $[s^{(i)}]$ directly, or by applying Construction 2. In the former case, clearly the terms and sums of terms of $[t^{(i)}]$ are known. In the latter case, the terms of $[t^{(i)}]$ are also known, since they are formed by summing terms from $[s^{(i)}]$, and the sums of terms of $[t^{(i)}]$ correspond to the double sums of terms of $[s^{(i)}]$, which are also known. Thus in each case the terms and sums of terms in $[t^{(i)}]$ are known.

Now consider applying the interleaving construction to $[t^{(i)}]$, to yield $[s^{(i+1)}]$. Clearly, the terms of $[s^{(i+1)}]$ may be found from the terms of $[t^{(i)}]$. By the nature of interleaving, a sum of terms in $[s^{(i+1)}]$ can be obtained from two expressions in sums of terms in $[t^{(i)}]$. The key to our method is to observe that the double sums of terms in $[s^{(i+1)}]$ may also be found, since by Remark 11 these correspond to sums of alternate terms in $[s^{(i+1)}]$, which are easy to find from sums of terms in $[t^{(i)}]$ since $[s^{(i+1)}]$ is formed by interleaving.

The above argument outlines a scheme by which the terms $L^{(i)}(j) = t_j^{(i)}$, which had previously to be stored, could be recursively computed instead. However, in the above we have greatly simplified the two constructions used, by ignoring the stages in which terms are inserted and removed from the sequences involved. It is necessary to form the sums and double sums above using the unsimplified forms of the two constructions. This is the purpose of Lemmas 12–14 below.

Lemma 12. *Let* $[s], [t], [a], [b], [d], k, v, n$ *and* m *be as in Construction* 7. *Let* $w = \mathrm{loc}([s], 1_{|2n|})$. *Assuming* k *is odd and that* $n \geqslant 3$, *then*

$$m = 2k - 2 + 2^{n-1}(2^n - 2) \tag{6}$$

$$= 2k + 2 + (2^{n-1} - 2)(2^n + 2) \tag{7}$$

$$v = \begin{cases} (2^n + 1 - k)(2^{n-1} - 1) & \text{if } k \equiv 1 \,(\mathrm{mod}\, 4), \text{ or} \\ (2^{n+1} + 3 - k)(2^{n-1} - 1) & \text{if } k \equiv 3 \,(\mathrm{mod}\, 4). \end{cases} \tag{8}$$

Also, if $v < m$ *then* $w = m + 3$ *and* $\sigma_w[s] = 0$, *or if* $v > m$ *then* $w = m + 1$ *and* $\sigma_w[s] = 1$.

Proof. Notice that $[a]_{k-1}^{n-1} = 1_{|n-1|}$ and $[b]_{k+1}^{n+1} = 1_{|n+1|}$, and that the unique position m of $1_{|2n-1|}$ in $[d]$ occurs when these windows of $[b]$ and $[a]$ are interleaved (uniqueness is assured by the proof of Construction 7). Since k is odd, $k+1$ and $k-1$ are both even, and we observe that digits with an even position in $[b]$ are always followed in $[d]$ by digits with an even position in $[a]$, thus the first digit of the window $1_{|2n-1|}$ in $[d]$ is obtained from the first digit of an instance of $1_{|n+1|}$ in $[b]$. Noting that $[a]$ and

[b] have least periods 2^n-2 and 2^n+2 respectively, we deduce that m is the unique solution modulo $2^{2n}-4$ of the simultaneous congruences

$$\frac{m}{2} \equiv k+1 (\mathrm{mod}\, 2^n+2)$$

and

$$\frac{m}{2} \equiv k-1 (\mathrm{mod}\, 2^n-2).$$

It is now simple to verify that the two equivalent formulae (6) and (7) are correct.

From Construction 7, it is sufficient to show that v satisfies

$$\frac{v}{2} \equiv k+1 (\mathrm{mod}\, 2^n+2)$$

and

$$\frac{v}{2} \equiv 0 (\mathrm{mod}\, 2^n-2).$$

Firstly, suppose $k=4c+1$ for some $c \in \mathbb{Z}$. Then, from Eq. (8) we have

$$\begin{aligned} v &= (2^n+1-k)(2^{n-1}-1) \\ &= (2^n-4c)(2^{n-1}-1) \\ &= (2^{n-1}-2c)(2^n-2) \\ &\equiv 0 (\mathrm{mod}\, 2^n-2) \end{aligned}$$

and, using the above, we also have

$$\begin{aligned} v &= (2^{n-1}-2c)(2^n-2) \\ &= (2^{n-1}-2c)(2^n+2) - 4(2^{n-1}-2c) \\ &\equiv 8c - 2^{n+1} (\mathrm{mod}\, 2^n+2) \\ &= 8c - 2(2^n+2) + 4 \\ &\equiv 2k+2 (\mathrm{mod}\, 2^n+2) \end{aligned}$$

as required. The case $k=4c+3$ is similar.

Now consider the process of constructing [s] from [d]. If $v<m$, then the digits '01' are inserted into [d] at a position before $1_{|2n-1|}$ appears in [d]. Together with the digit '0' being inserted into [d] at position 0, this gives $w=m+3$. That $w=m+1$ when $v>m$ is also clear.

Notice from Eq. (6) that m is even. We have

$$\begin{aligned} \sigma_m[d] &= \xi_e([d], m/2) + \xi_o([d], m/2) \\ &= \sigma_{m/2}[b] + \sigma_{m/2}[a]. \end{aligned} \tag{9}$$

Since [a] has least period 2^n-2 and generalised weight 1, and from its relation to [t], we have

$$\sigma_\ell[a] = \begin{cases} \sigma_{\ell+1}[t] & \text{if } 1 \leqslant \ell \leqslant k-1, \\ \sigma_{\ell+2}[t] - 1 & \text{if } k \leqslant \ell \leqslant 2^n-2, \text{ or} \\ \sigma_{\ell \bmod (2^n-2)}[a] + \lfloor \frac{\ell}{2^n-2} \rfloor & \text{if } \ell \geqslant 2^n-1. \end{cases} \tag{10}$$

Taking m from Eq. (6), and substituting $m/2$ in place of ℓ in Eq. (10) gives

$$\begin{aligned}\sigma_{m/2}[a] &= \sigma_{k-1}[a] + 2^{n-2} \\ &= \sigma_{k-1}[a] \\ &= \sigma_k[t], \end{aligned} \tag{11}$$

using the condition $n \geqslant 3$. Similarly,

$$\sigma_\ell[b] = \begin{cases} \sigma_{\ell-1}[t] & \text{if } 1 \leqslant \ell \leqslant k+1, \\ \sigma_{\ell-2}[t] + 1 & \text{if } k+2 \leqslant \ell \leqslant 2^n + 2, \text{ or} \\ \sigma_{\ell \bmod (2^n+2)}[b] + \lfloor \frac{\ell}{2^n+2} \rfloor & \text{if } \ell \geqslant 2^n + 3. \end{cases} \tag{12}$$

Taking m from Eq. (7) and substituting $m/2$ for ℓ in the above gives

$$\begin{aligned}\sigma_{m/2}[b] &= \sigma_{k+1}[b] + 2^{n-2} - 1 \\ &= \sigma_k[t] + 1. \end{aligned} \tag{13}$$

Combining Eqs. (9), (11) and (13) gives $\sigma_m[d] = 1$. It is then simple to see that if $v > m$ then $\sigma_w[s] = \sigma_m[d] = 1$ and that if $v < m$ then $\sigma_w[s] = \sigma_m[d] + 1 = 0$, as required. □

Lemma 13. *We adopt the notation and assumptions of Lemma* 12.

(i) *For any i in the range $1 \leqslant i \leqslant 2^{2n}$ the value of $\sigma_i[s]$ can be found using k and at most two values $\sigma_j[t]$ and $\sigma_{j'}[t]$, where j and j' lie in the range $1 \leqslant j, j' \leqslant 2^n$.*

(ii) *For any i in the range $2 \leqslant i \leqslant 2^{2n}$ the value of $\eta_i[s]$ can be found using k and at most one value $\sigma_j[t]$, where j lies in the range $1 \leqslant j \leqslant 2^n$.*

Proof. For part (1), if $v < m$ we have

$$\sigma_i[s] = \begin{cases} \sigma_{i-1}[d] & \text{if } 1 \leqslant i \leqslant v+1, \\ \sigma_v[d] + 1 & \text{if } i = v+2, v+3, \\ \sigma_{i-3}[d] + 1 & \text{if } v+4 \leqslant i \leqslant m+3, \\ \sigma_m[d] + 2 & \text{if } i = m+4, \text{or} \\ \sigma_{i-4}[d] + 2 & \text{if } m+5 \leqslant i \leqslant 2^{2n}. \end{cases}$$

Similar expressions hold when $v > m$.

The value $\sigma_\ell[d]$ for all $\ell \geqslant 1$ may be computed from the information described in the statement of the Lemma, by using Eqs. (10) and (12) together with $\sigma_\ell[d] = \sigma_{\lceil \ell/2 \rceil}[b] + \sigma_{\lfloor \ell/2 \rfloor}[a]$.

For part (2), applying Remark 11 gives

$$\eta_i[s] = \begin{cases} \xi_e([s], i/2) & \text{for } i \text{ even, or} \\ \xi_0([s], (i-1)/2) & \text{for } i \text{ odd.} \end{cases}$$

Notice from Lemma 12 that v and m are both even, and easily computed from k. If $v < m$ then

$$\xi_e([s],\ell) = \begin{cases} \xi_o([d],\ell-1) & \text{if } 1 \leqslant \ell \leqslant v/2+1, \\ \xi_o([d],v/2) & \text{if } \ell = v/2+2, \\ \xi_o([d],\ell-2) & \text{if } v/2+3 \leqslant \ell \leqslant m/2+2, \text{ or} \\ \xi_o([d],m/2)+\xi_e([d],\ell-2) & \\ \quad -\xi_e([d],m/2) & \text{if } m/2+3 \leqslant \ell \leqslant 2^{2n-1}. \end{cases}$$

Similarly,

$$\xi_o([d],\ell) = \begin{cases} \xi_e([d],\ell) & \text{if } 1 \leqslant \ell \leqslant v/2, \\ \xi_e([d],v/2)+1 & \text{if } \ell = v/2+1, \\ \xi_e([d],\ell-1)+1 & \text{if } v/2+2 \leqslant \ell \leqslant m/2+1, \\ \xi_e([d],\ell-2)+2 & \text{if } \ell = m/2+2, \text{ or} \\ \xi_e([d],m/2)+2+\xi_o([d],\ell-2) & \\ \quad -\xi_o([d],m/2) & \text{if } m/2+3 \leqslant \ell \leqslant 2^{2n-1}. \end{cases}$$

Similar expressions may be derived in the case $v > m$. Expressions concerning $\xi_e([d],m/2)$ and $\xi_o([d],m/2)$ in each of the above equations can be simplified by substituting $\xi_o([d],m/2) = \sigma_{m/2}[a] = \sigma_k[t]$ (applying Eq. (11)) and $\xi_e([d],m/2) = \sigma_{m/2}[b] = \sigma_k[t]+1$ (applying Eq. (13)). Since $\sigma_k[t]$ appears either not at all or twice in each of the expressions for $\xi_o([d],\ell)$ and $\xi_e([s],\ell)$ thus arising the result is not dependent on $\sigma_k[t]$. In each case we are left with an expression for $\eta_i[s]$ in terms of at most one of $\xi_o([d],\ell)$ or $\xi_e([d],\ell)$, for some ℓ in the range $1 \leqslant \ell \leqslant 2^{2n-1}-1$. From the definition of $[d]$ we deduce that $\xi_o([d],\ell) = \sigma_\ell[a]$ and $\xi_e([d],\ell) = \sigma_\ell[b]$. By applying either Eq. (10) or Eq. (12), as appropriate, we obtain $\eta_i[s]$ in terms of k and $\sigma_j[t]$, where j lies in the range $1 \leqslant j \leqslant 2^n$. □

Lemma 14. *Let n be an integer, $n \geqslant 2$, let $[s]$ be a de Bruijn sequence of span n, let $k = \mathrm{loc}([s], 1_{|n|})$ and let $[t]$ be given by $[t] = \Phi[s]$. Let $[s']$ and $[\hat{s}]$ be the intermediate sequences formed during the construction of $[t]$ from $[s]$.*

(i) *For any i in the range $0 \leqslant i \leqslant 2^{n+1}-1$ the value of t_i can be found using k and at most one value $\sigma_j[s]$, for some j in the range $1 \leqslant j \leqslant 2^n-1$.*
(ii) *For any i in the range $1 \leqslant i \leqslant 2^{n+1}$ the value of $\sigma_i[t]$ can be found using k, $\eta_{2^n}[s]$ and at most one value $\eta_j[s]$, for some j in the range $2 \leqslant j \leqslant 2^n$.*

Proof. We have $[s'] = \chi_k[s]$, $s_k = 1$ and $[\hat{s}] = D_0^{-1}[s']$, thus

$$\hat{s}_i = \begin{cases} \sigma_i[s] & \text{if } 0 \leqslant i \leqslant k, \\ \sigma_{i+1}[s]-1 & \text{if } k+1 \leqslant i \leqslant 2^n-2, \text{ or} \\ \hat{s}_{i-(2^n-1)}+1 & \text{if } 2^n-1 \leqslant i \leqslant 2^{n+1}-3. \end{cases}$$

Considering the final stage of Construction 2, since the alphabet size 2 is at most equal to the span of $[s]$, letting $p = 2^n - 1 + k$, we have

$$t_i = \begin{cases} \hat{s}_i & \text{if } 0 \leqslant i \leqslant p+1, \text{ or} \\ \hat{s}_{i-2} & \text{if } p+2 \leqslant i \leqslant 2^{n+1}-1. \end{cases}$$

This argument gives an explicit means of computing t_i as required by part (1) of the Lemma.

For part (2), letting $p = 2^n - 1 + k$ and from the relation of $[t]$ to $[\hat{s}]$ we have

$$\sigma_i[t] = \begin{cases} \sigma_i[\hat{s}] & \text{if } 1 \leqslant i \leqslant p+2, \text{ or} \\ \sigma_{i-2}[\hat{s}] + 1 & \text{if } p+3 \leqslant i \leqslant 2^{n+1}. \end{cases}$$

It is clear that since $[\hat{s}] = D_0^{-1}[s']$, we have $\sigma_\ell[\hat{s}] = \eta_\ell[s']$ for all ℓ in the range $2 \leqslant \ell \leqslant 2^{n+1} - 2$, and that $\sigma_1[\hat{s}] = 0$. Next, since $[s'] = \chi_k[s]$ and $[s]$ has least period 2^n we have

$$\eta_\ell[s'] = \begin{cases} \eta_\ell[s] & \text{if } 2 \leqslant \ell \leqslant k+1, \\ \eta_{\ell+1}[s] - (\ell - k - 1) & \text{if } k+2 \leqslant \ell \leqslant 2^n + k + 1, \text{ or} \\ \eta_{\ell+2}[s] - (\ell - k - 1) & \\ \quad -(\ell - k - 1 - 2^n) & \text{if } 2^n + k + 2 \leqslant \ell \leqslant 2^{n+1} - 2. \end{cases}$$

Finally, if $2^n + 2 \leqslant \ell \leqslant 2^{n+1}$ then $\eta_\ell[s] = \eta_{\ell-2^n}[s] + \eta_{2^n+1}[s] + (\ell - 2^n)\mathrm{wt}([s])$. However, notice that the last term here is zero since $\mathrm{wt}([s]) = 0$ by Lemma 1. Also, since $[s]$ has its maximal run of consecutive zeroes at position 0, we must have $s_{2^n-1} = 1$. Thus, we may substitute $\eta_{2^n+1}[s] = \eta_{2^n}[s] + \sigma_{2^n}[s] = \eta_{2^n}[s] + \mathrm{wt}([s]) - 1 = \eta_{2^n}[s] + 1$. Together, these relationships show how the computation described in the statement of the lemma may be performed, as required. □

We now make a slight change to our construction, involving the initial sequence that the construction uses. The main reason for doing this is to ensure that the conditions $n \geqslant 3$ and that k is odd in Lemma 12 are satisfied when this Lemma is later employed.

Construction 15 (The binary hybrid construction). *To begin, we define binary de Bruijn sequences* $[\kappa^1]$ *and* $[\kappa^2]$, *with spans* 3 *and* 4 *respectively, by*

$$[\kappa^1] = [00011101],$$
$$[\kappa^2] = [0000100111101011].$$

Suppose that we wish to construct a span n de Bruijn sequence, where n is any integer greater than or equal to 4. *Let* $d_n = \lfloor 1 + \log_2 n \rfloor$ *be the number of digits in the binary representation of n and denote this representation by* $n_1 n_2 \cdots n_d$, *where the most significant digit is to the left* (*so* $n = \sum_{i=1}^{d} n_i 2^{d-i}$). *Necessarily,* $n_1 = 1$ *and* $d \geqslant 3$.

We define the sequence $[s^{(3)}]$ *according to the value of* n_2, *by*

$$[s^{(3)}] = \begin{cases} \Upsilon[\kappa^1] & \text{if } n_2 = 1, \text{ or} \\ [\kappa^2] & \text{if } n_2 = 0. \end{cases}$$

We then inductively define the sequences $[t^{(i)}]$ *for* $3 \leqslant i \leqslant d$ *and* $[s^{(i)}]$ *for* $4 \leqslant i \leqslant d$ *by*

$$[t^{(i)}] = \begin{cases} [s^{(i)}] & \text{if } n_i = 0, \text{ or} \\ \Phi[s^{(i)}] & \text{if } n_i = 1, \text{ and} \end{cases} \tag{14}$$

$$[s^{(i+1)}] = \Upsilon[t^{(i)}]. \tag{15}$$

As before, we define $m_i = \sum_{j=1}^{i} n_j 2^{i-j}$ for all i in the range $1 \leqslant i \leqslant d$ and note that $[t^{(i)}]$ and $[s^{(i)}]$ are de Bruijn sequences of span m_i and $2m_{i-1}$ respectively, for each i in the range $3 \leqslant i \leqslant d$.

For each i in the range $3 \leqslant i \leqslant d$ we define integers $k^{(i)}$ and $k'^{(i)}$ by $k'^{(i)} = \mathrm{loc}([s^{(i)}], 1_{|2m_{i-1}|})$ and $k^{(i)} = \mathrm{loc}([t^{(i)}], 1_{|m_i|})$. We also define $[a^{(i)}], [b^{(i)}]$ and $[d^{(i)}]$ for each i in the range $3 \leqslant i \leqslant d-1$ to be those intermediate sequences obtained when applying Construction 7 to $[t^{(i)}]$ to obtain $[s^{(i+1)}]$, and define integers $v^{(i)}$ and $m^{(i)}$ by $v^{(i)} = \mathrm{loc}([d^{(i)}], X_{2m_i-2})$ and $m^{(i)} = \mathrm{loc}([d^{(i)}], 1_{|2m_i-1|})$.

Lemma 16. *With the above notation*

$$k'^{(3)} = \begin{cases} 7 & \text{if } n_2 = 0, \text{ or} \\ 29 & \text{if } n_2 = 1, \end{cases}$$

and the remaining values of $k'^{(i)}$, *together with the values of* $v^{(i)}$, $m^{(i)}$ *and* $k^{(i)}$ *for appropriate ranges of* i *may be found from the following recurrences*:

$$k^{(i)} = \begin{cases} k'^{(i)} & \text{if } n_i = 0, \text{ or} \\ 2^{2m_{i-1}} - 1 & \text{if } n_i = 1, \end{cases}$$

$$m^{(i)} = 2k^{(i)} - 2 + 2^{2m_i - 1}(2^{2m_i} - 2),$$

$$v^{(i)} = \begin{cases} (2^{m_i} + 1 - k^{(i)})(2^{2m_i-1} - 1) & \text{if } k^{(i)} \equiv 1 \pmod 4, \text{or} \\ (2^{m_i+1} + 3 - k^{(i)})(2^{2m_i-1} - 1) & \text{if } k^{(i)} \equiv 3 \pmod 4 \end{cases}$$

$$k'^{(i+1)} = \begin{cases} m^{(i)} + 3 & \text{if } v^{(i)} < m^{(i)}, \text{or} \\ m^{(i)} + 1 & \text{if } v^{(i)} > m^{(i)}. \end{cases}$$

Moreover, $k^{(i)}$ *and* $k'^{(i)}$ *are always odd, and* $m^{(i)}$ *and* $v^{(i)}$ *are always even.*

Proof. Clearly, if $n_2 = 0$ then $k'^{(3)} = \mathrm{loc}([\kappa^2], 1_{|4|}) = 7$. If $n_2 = 1$ then that $k'^{(3)} = 29$ follows from the fact that $\mathrm{loc}([\kappa^1], 1_{|3|}) = 3$ by applying Lemma 12. Notice that both values for $k^{(3)}$ are odd.

The formula for $k^{(i)}$ is trivial when $n_i = 0$. When $n_i = 1$ it follows from an argument like that used to derive Eq. (2). Notice that these values too are odd, provided that $k'^{(i)}$ is odd.

The formulae for $m^{(i)}$, $v^{(i)}$ and $k'^{(i+1)}$ follow from Lemma 12, provided that $k^{(i)}$ is always odd. Notice that the expressions for $m^{(i)}$ and $v^{(i)}$ are even, implying that $k'^{(i+1)}$ is odd. The result follows for all i by induction. □

Theorem 17. *Let n be any integer, with $n \geqslant 4$. Then there exists a binary de Bruijn sequence of span n, formed using Construction* 15, *that may be decoded using an algorithm that has a computational complexity of* $O(n \log n)$ *integer operations and requires* $O(1)$ *bits of storage.*

Proof. The decoding technique is like that described in Section 5 for sequences over arbitrary alphabets. It suffices to show that all of the information which had previously to be stored may now be computed instead, given tables of size $O(1)$ bits, and to consider the computational complexity of the resulting procedure.

Lemma 16 gives relations that allow the $m^{(i)}$, $v^{(i)}$, $k^{(i)}$ and $k'^{(i)}$ to be computed inductively for each i, using $O(i)$ integer operations. To show how the $L^{(i)}(j) = t_j^{(i)}$ may be computed, we prove the following proposition: for each i in the range $4 \leqslant i \leqslant d$ we may compute the value of $\sigma_j[t^{(i)}]$, for any j in the range $1 \leqslant j \leqslant 2^{m_i}$, using at most two values $\sigma_\ell[t^{(i-1)}]$ and $\sigma_{\ell'}[t^{(i-1)}]$ for some ℓ and ℓ' in the range $1 \leqslant \ell, \ell' \leqslant 2^{m_{i-1}}$.

Suppose that $n_i = 0$. Then $[t^{(i)}] = [s^{(i)}]$, and so the required sum $\sigma_j[t^{(i)}]$ is simply equal to $\sigma_j[s^{(i)}]$. Since $[s^{(i)}] = \Upsilon[t^{(i-1)}]$, Lemma 13, part (1) shows how this sum may be expressed in terms of at most two sums $\sigma_\ell[t^{(i-1)}]$ and $\sigma_{\ell'}[t^{(i-1)}]$.

Now suppose that $n_i = 1$. Then we have $[t^{(i)}] = \Phi[s^{(i)}]$, and so we may use Lemma 14, part (2), to express the required sum $\sigma_j[t^{(i)}]$ in terms of at most two double sums $\eta_{2^{2m_i-1}}[s^{(i)}]$ and $\eta_{j'}[s^{(i)}]$. Lemma 13, part (2), shows how each of these may be computed from a sum $\sigma_\ell[t^{(i-1)}]$, as required.

The proof of the proposition is complete. By storing the values $\sigma_j[t^{(3)}]$ in a table (of size $O(1)$ bits), the proposition gives rise to an inductive proof of the fact that we may calculate any expression of the form $\sigma_j[t^{(i)}]$ recursively.

We need now remark only that part (1) of Lemma 14 shows how the required values $L^{(i)}(j) = t_j^{(i)}$ may be computed from a sum $\sigma_\ell[s^{(i)}]$, and that this sum may be computed from at most two sums $\sigma_\ell[t^{(i-1)}]$ and $\sigma_{\ell'}[t^{(i-1)}]$ using part (1) of Lemma 13. These sums may be computed recursively by the method just described, and so the need to store the values $L^{(i)}(j)$ has been eliminated.

The recursive step in computing $L^{(i)}(j)$ at most doubles the number of expressions that need to be computed at the next recursive step. Also, from the descriptions of the individual calculations in the earlier Lemmas, the number of operations in each individual recursive step may be bounded above by a constant, and so computing $L^{(i)}(j)$ requires $O(1 + 2 + 4 + \cdots + 2^{i-3}) = O(2^{i-2})$ integer operations.

Considering the worst case, where $n_i = 1$ for all $i = 3, 4, \ldots, d$ and summing the computational requirements described above over this range of i, we obtain a total computational requirement of $O(\sum_{i=3}^{d}(i + 2^{i-2})) = O(d^2 + 2^{d-1}) = O(n)$ integer operations. This is dominated by the $O(n \log n)$ operations already required by the algorithm, obtained by substituting $c = 2$ into Theorem 10. □

Notice that the storage requirements for the above procedure, as well as having an asymptotic size of O(1) bits, are very small in absolute terms. It is common for a small, fixed storage requirement such as this to be considered an integral part of the decoding algorithm, and for the algorithm then to be classed as 'storage-free'. This is the first time, for any fixed alphabet size, that a construction for de Bruijn sequences of arbitrary span has been exhibited, that allows computationally efficient, storage-free decoding.

7. Composite alphabets

In this section we give a construction for easily decoded de Bruijn sequences over alphabets of composite size.

Let y and z be integers, with $y,z \geqslant 2$, and let $[r]$ and $[s]$ be sequences of least periods p and q over y-ary and z-ary alphabets $\mathscr{A}$ and $\mathscr{B}$, respectively. We define the *product* of the sequences $[r]$ and $[s]$, denoted $[r] \times [s]$ to be the sequence $[t]$ of least period $\mathrm{lcm}(p,q)$ over the alphabet $\mathscr{A} \times \mathscr{B}$ of size yz given by $t_i = (r_i, s_i)$ for all $i \in \mathbb{Z}$.

The following sequences are a special case of the perfect multi-factors of Mitchell [3, Definition 2.1].

Definition 18. Let z,m and n be positive integers, with $z \geqslant 2$. A z-ary sequence $[s]$ of least period mz^n is an (n,m) *spaced de Bruijn sequence* (hereafter an (n,m)-SDB sequence) if and only if for each k in the range $0 \leqslant k \leqslant m-1$, every possible n-tuple appears in $[s]$ at a unique position i with $i \equiv k \pmod m$.

We define the decoding problem a (n,m)-SDB sequence $[s]$ over a z-ary alphabet $\mathscr{B}$ as follows: given any $v \in \mathscr{B}^n$ and any integer f in the range $0 \leqslant f \leqslant m-1$, the decoding problem for $[s]$ is to determine the unique integer g in the range $0 \leqslant g \leqslant z^n - 1$ such that $[s]^n_{gm+f} = v$. We write $g = \mathrm{loc}([s], v, f)$.

The following result is a specialisation of [3, Theorem 5.2].

Theorem 19. *Let $[r]$ be a span n de Bruijn sequence over a y-ary alphabet $\mathscr{A}$ and let $[s]$ be a (n, y^n)-SDB sequence over a z-ary alphabet $\mathscr{B}$. Then the sequence $[t] = [r] \times [s]$ is a yz-ary de Bruijn sequence of span n over $\mathscr{A} \times \mathscr{B}$.*

Now consider the decoding problem for $[t]$. Given $w \in (\mathscr{A} \times \mathscr{B})^n$, we may project w onto $\mathscr{A}$ and $\mathscr{B}$ to get n-tuples $u \in \mathscr{A}^n$ and $v \in \mathscr{B}^n$ respectively. Now we solve one instance of the decoding problem for $[r]$, letting $f = \mathrm{loc}([r], u)$, and then solve one instance of the decoding problem for $[s]$, letting $g = \mathrm{loc}([s], v, f)$. It follows from the definitions that w occurs in $[t]$ at position $gy^n + f$.

We shall require some definitions and notation.

Let n and z be positive integers, with $z \geqslant 2$. We define the map $\psi_{z,n} : \mathbb{Z}_{z^n} \to \mathbb{Z}_z^n$ as returning the n-tuple corresponding to the n digit base z representation of the integer

x, with the least significant digit positioned to the right. Notice that $\psi_{z,n}$ is a bijection, with inverse $\psi_{z,n}^{-1}$. Let $\mathscr{C}$ denote the $[n+1,n,2]$ generalised parity check code over $\mathbb{Z}_z$, formed by taking those $(n+1)$-tuples whose coordinates sum to zero in $\mathbb{Z}_z$. We define the map $\theta_{z,n} : \mathbb{Z}_{z^n} \to \mathscr{C}$ as returning, for any $x \in \mathbb{Z}_{z^n}$, the unique codeword in $\mathscr{C}$ whose rightmost n digits are equal to $\psi_{z,n}(x)$.

Let $w=(w_{n-1}, w_{n-2}, \ldots, w_0)$ be an arbitrary n-tuple. We define the left-shift operator E on n-tuples by $E(w) = (w_{n-2}, w_{n-3}, \ldots, w_0, w_{n-1})$. We also extend our definition of the [ ] operator. We have used these brackets to define a sequence from a generating cycle, for example writing $[s] = [012]$. For an n-tuple w as defined above we write $[w]$ to denote the sequence defined by $[w] = [w_{n-1}, w_{n-2}, \ldots, w_0]$.

Let $\mathscr{A}$ denote any alphabet, and let p and q be positive integers and let $u = (u_{p-1}, u_{p-2}, \ldots, u_0) \in \mathscr{A}^p$ and $v = (v_{q-1}, v_{q-2}, \ldots, v_0) \in \mathscr{A}^q$. We define the *concatenation* of u and v, denoted $u|v$, by

$$u|v = (u_{p-1}, u_{p-2}, \ldots, u_0, v_{q-1}, v_{q-2}, \ldots, v_0) \in \mathscr{A}^{p+q}.$$

Since this operator is associative, we do not need parentheses in expressions with multiple concatenations. This allows us to define u^t as denoting the t-fold concatenation of u with itself, for all positive integers t. For any n-tuple u, we define u^0 to be the 0-tuple (the identity with respect to concatenation).

Recall that we write $[s]_i^n$ to denote the n-window of $[s]$ beginning with the digit s_i.

Lemma 20. *Let n and z be integers, with $n \geqslant 1$ and $z \geqslant 2$. Let x be any element of $\mathbb{Z}_{z^n}$. Then for each i in the range indicated, the functions*

$$\begin{aligned}
&A_i : x \mapsto [\psi_{z,n}(x)|\psi_{z,n}(x)]_i^n \quad \text{for } 0 \leqslant i < n,\\
&B_i : x \mapsto [\psi_{z,n}(x)|\psi_{z,n}(x+1)]_i^n \quad \text{for } 0 \leqslant i < n,\\
&C_i : x \mapsto [\psi_{z,n}(x)|\theta_{z,n}(x)]_i^n \quad \text{for } 0 \leqslant i < n,\\
&D_i : x \mapsto [\theta_{z,n}(x)|\psi_{z,n}(x+1)]_i^n \quad \text{for } 0 \leqslant i \leqslant n,\\
&E_i : x \mapsto [\theta_{z,n}(x)|\theta_{z,n}(x)]_i^n \quad \text{for } 0 \leqslant i \leqslant n, \text{ and}\\
&F_i : x \mapsto [\theta_{z,n}(x)|\theta_{z,n}(x+1)]_i^n \quad \text{for } 0 \leqslant i \leqslant n
\end{aligned}$$

are each bijections from $\mathbb{Z}_{z^n}$ to $\mathbb{Z}_z^n$. Moreover, they each have an inverse that may be computed using at most $O(n)$ integer operations.

Proof. We give a proof for the function F_i only, the remaining cases being similar. We will describe a sequence of operations to find x given its image. Since $|\mathbb{Z}_{z^n}| = |\mathbb{Z}_z^n|$, this implies that F_i is a bijection.

Let $v=(v_{n-1}, v_{n-2}, \ldots, v_0) = F_i(x)$ and let $w=(w_n, w_{n-1}, \ldots, w_0) = \theta_{z,n}(x+1)$, where x is any element of $\mathbb{Z}_{z^n}$. If $i=0$ or $i=1$ then n digits of w are known, and the last easily found since the digits of w sum to zero in $\mathbb{Z}_c$. We have $x = \psi_{z,n}^{-1}(w_{n-1}, w_{n-2}, \ldots, 0)$.

Suppose $i \geqslant 2$. The $(n-i+1)$-tuple $u=(v_{n-1}, v_{n-2}, \ldots, v_{i-1})$ represents the $n-i+1$ least significant digits in $\psi_{z,n}(x)$. It is simple to compute $u' = \psi_{z,n-i+1}(\psi_{z,n-i+1}^{-1}(u)+1)$ — this is simply adding 1 to the value represented in base z by u, modulo z^{n-i+1}. Then u' consists of the $n-i+1$ least significant digits of $\psi_{z,n}(x+1)$, namely $(w_{n-i}, w_{n-i-1}, \ldots, w_0)$.

Together with $(v_{i-2}, v_{i-3}, \ldots, v_0) = (w_n, w_{n-1}, \ldots, w_{n-i+2})$, the only digit of w we do not have is w_{n-i+1}. By the manner used in the case $i = 0$ or 1, this digit is easily found, and the value of $x + 1$ and thus x thus recovered.

In each case, at most $O(n)$ integer operations are used. □

Construction 21. *Let n, z, k and ℓ be integers, with $k, \ell \geqslant 0$, $n \geqslant 1$, $z \geqslant 2$ and k, ℓ not both zero. For each $x \in \mathbb{Z}_{z^n}$, we define the $(nk + (n+1)\ell)$-tuple d_x by*

$$d_x = (\psi_{z,n}(x))^k | (\theta_{z,n}(x))^\ell. \tag{16}$$

We construct the sequence $[s]$ by

$$[s] = [d_0 | d_1 | \cdots | d_{z^n - 1}]. \tag{17}$$

Lemma 22. *The sequence $[s]$ constructed in the manner of Construction 21 is a z-ary (n, m)-SDB sequence, where $m = nk + (n+1)\ell$. Moreover, the decoding problem for $[s]$ may be solved using at most $O(n)$ integer operations.*

Proof. It is clear that $[s]$ has least period mz^n, as required. Let f be any integer in the range $0 \leqslant f < m$. We consider a number of distinct cases. These are distinguished by f, k and ℓ, and apply for all g in the range $0 \leqslant g < z^n$. In each, we shall express $[s]^n_{gm+f}$ in terms of g, using one of the functions described in Lemma 20. Since these functions are bijections from $\mathbb{Z}_{z^n}$ to $\mathbb{Z}_z^n$, this will imply that every element of $\mathbb{Z}_z^n$ appears exactly once in $[s]$, at a position congruent to f modulo m.

Case $\ell = 0$, $0 \leqslant f < n(k-1)$: $[s]^n_{gm+f} = A_{f \bmod n}(g)$.

Case $\ell = 0$, $n(k-1) \leqslant f < nk$: $[s]^n_{gm+f} = B_{f \bmod n}(g)$.

Henceforth we assume $\ell \neq 0$.

Case $k \neq 0$ and $f < n(k-1)$: $[s]^n_{gm+f} = A_{f \bmod n}(g)$.

Case $k \neq 0$ and $n(k-1) \leqslant f < nk$: $[s]^n_{gm+f} = C_{f \bmod n}(g)$.

Case $k \neq 0$ and $nk \leqslant f < nk + (n+1)(\ell - 1)$: $[s]^n_{gm+f} = E_{(f-nk) \bmod (n+1)}(g)$.

Case $k \neq 0$ and $nk + (n+1)(\ell - 1) \leqslant f < m$: $[s]^n_{gm+f} = D_{(f-nk) \bmod (n+1)}(g)$.

Case $k = 0$ and $f < (n+1)(\ell - 1)$: $[s]^n_{gm+f} = E_{f \bmod (n+1)}(g)$.

Case $k = 0$ and $(n+1)(\ell - 1) \leqslant f$: $[s]^n_{gm+f} = F_{f \bmod (n+1)}(g)$.

It remains to observe that since each of the functions $A_i, \ldots, F_i$ have an inverse that may be computed in $O(n)$ integer operations, and since each of the above cases may be distinguished at least as easily, the decoding problem for $[s]$ may be solved using $O(n)$ integer operations as stated in the lemma. □

We now address the following naturally occurring question: given n and z, for what values m does Construction 21 allow the construction of a z-ary (n, m)-SDB sequence?

Lemma 23. *Let n and m be positive integers. Then Construction 21 can be used to form an (n, m)-SDB sequence over an alphabet of arbitrary size if and only if*

$$m \bmod n \leqslant \left\lfloor \frac{m}{n} \right\rfloor. \tag{18}$$

Proof. Suppose that n and m satisfy the inequality (18). Let $\ell = m \bmod n$ and let $k = \lfloor m/n \rfloor - \ell$. The hypothesis ensures that $k \geqslant 0$. Notice that

$$\begin{aligned} m &= n \times \left\lfloor \frac{m}{n} \right\rfloor + (m \bmod n) \\ &= n(k+\ell) + \ell \\ &= nk + (n+1)\ell, \end{aligned}$$

so by Lemma 22 the sequence $[s]$ constructed using these values of k and ℓ in Construction 21 is an (n,m)-SDB sequence as required.

Now suppose that $[s]$ is an (n,m)-SDB sequence formed using Construction 21. For some integers $k, \ell \geqslant 0$, not both zero, we have $m = nk + (n+1)\ell = n(k+\ell) + \ell$. Thus

$$\begin{aligned} m \bmod n &= \ell \bmod n \\ &\leqslant k + \ell \\ &\leqslant \left\lfloor \frac{m}{n} \right\rfloor \end{aligned}$$

as required. The last stage follows since $m = nk + (n+1)\ell$ implies that $k + \ell \leqslant m/n$ and from the fact that k and ℓ are integers. □

Theorem 24. *Let n and c be integers, with $n, c \geqslant 2$ and with c composite. Let y be any non-trivial factor of c, and let $[r]$ be any y-ary span n de Bruijn sequence. Define $z = c/y$. Then there exists a c-ary span n de Bruijn sequence $[t] = [r] \times [s]$, where $[s]$ is a z-ary (n, y^n)-SDB sequence constructed in the manner of Construction 21. Moreover, the decoding problem for $[t]$ may be solved using $O(n)$ integer operations, together with one decoding of the sequence $[r]$.*

Proof. Since $[r]$ is a y-ary de Bruijn sequence of span n, we know that it has least period y^n. It is simple to show that for any $n, y \geqslant 2$, we have $y^n \bmod n \leqslant \lfloor y^n/n \rfloor$. By Lemmas 22 and 23 we may use Construction 21 to form a z-ary (n, y^n)-SDB sequence $[s]$. By Theorem 19, the sequence $[t] = [r] \times [s]$ is a span n de Bruijn sequence over an alphabet of size $c = yz$.

The decoding of $[t]$ may be obtained in the manner described earlier in this Section. The complexity follows from Lemma 22. □

Example 25. *Suppose we wish to construct a c-ary de Bruijn sequence of span n, with $c = 15$ and $n = 2$. Factoring c, let $y = 3$ and $z = 5$. Let $[r] = [002212011]$, a de Bruijn sequence of span 2 over $\mathbb{Z}_3$. Since the least period of $[r]$ is $y^n = 9$, we seek to construct a $(2,9)$-SDB sequence $[s]$ over $\mathbb{Z}_5$. Following Lemma 22, we require integers k and ℓ such that $9 = 2k + 3\ell$. We may choose either $k = 0$ and $\ell = 3$, or $k = 3$ and $\ell = 1$; in this example we opt for the latter. From Eq. (17), $[s]$ is formed*

by concatenating the 9*-tuples* d_x *for* x *running from* 0 *to* $z^n - 1 = 24$, *where* d_x *is as given in Eq.* (16). *That is*

$$
\begin{aligned}
[s] = [&00\ 00\ 00\ 000\ 01\ 01\ 01\ 401\ 02\ 02\ 02\ 302\ 03\ 03\ 03\ 203\ 04\ 04\ 04\ 104\\
&10\ 10\ 10\ 410\ 11\ 11\ 11\ 311\ 12\ 12\ 12\ 212\ 13\ 13\ 13\ 113\ 14\ 14\ 14\ 014\\
&20\ 20\ 20\ 320\ 21\ 21\ 21\ 221\ 22\ 22\ 22\ 122\ 23\ 23\ 23\ 023\ 24\ 24\ 24\ 424\\
&30\ 30\ 30\ 230\ 31\ 31\ 31\ 131\ 32\ 32\ 32\ 032\ 33\ 33\ 33\ 433\ 34\ 34\ 34\ 334\\
&40\ 40\ 40\ 140\ 41\ 41\ 41\ 041\ 42\ 42\ 42\ 442\ 43\ 43\ 43\ 343\ 44\ 44\ 44\ 244].
\end{aligned}
$$

(*The spaces serve no purpose other than to clarify the structure.*) *The product of* $[r]$ *and* $[s]$ *is then a span* 2 *de Bruijn sequence over* $\mathbb{Z}_3 \times \mathbb{Z}_5$.

Decoding any span n de Bruijn sequence will always require $\Omega(n)$ operations, since this is the number of operations required to process the input. Thus the decoding procedure for $[t]$ described above has asymptotically optimal complexity, except for the effort required to perform a single decoding of the de Bruijn sequence $[r]$. Also, we impose no restriction on the techniques used to construct and decode $[r]$ — any current or future technique may be employed. Thus we argue that the method of this Section represents a powerful tool for constructing a wide class of easily decoded de Bruijn sequences.

As well as being used to form spaced de Bruijn sequences, Construction 21 may be generalised to give a class of Mitchell's perfect multi-factors [3], including those perfect multi-factors arising from [3, Construction 3.5]. Details may be found in [7].

8. Conclusion

As a final comment for our work on decoding de Bruijn sequences, we note that the sequence $[r]$ in Theorem 24 may be constructed by the means described in Constructions 9 and 15. Combining our earlier results gives:

Theorem 29. *Let* n *and* c *be positive integers, with* $c \geqslant 2$, *and let* a *and* b *be positive integers such that* $n = a2^b$ *with* a *odd. Then there exists a* c*-ary de Bruijn sequence* $[t]$ *possessing a decoding algorithm with the following requirements*:

(1) *for* c *even, the storage requirement is bounded by a constant and the computational requirement by* $O(n \log n)$ *integer operations, or*
(2) *for* c *odd, for any integer* y *with* $y|c$ *and* $y \geqslant 2$ *the storage requirement is* $O(y^a)$ *bits and the computational requirement is* $O(n \log n(1 + \log y))$ *integer operations.*

Part (1) of Theorem 26 in effect states that the problem of constructing easily decoded de Bruijn sequences is 'solved' in the case of arbitrary even alphabet size and arbitrary span. Examining part (2) shows that the decoding techniques in this paper are at their least efficient when the alphabet size c is an odd prime and the span n is odd.

Alternative techniques for constructing and decoding de Bruijn sequences, quickly and without the use of large storage tables, remain a desirable goal in these cases.

Acknowledgements

The author is grateful to Professor Chris Mitchell for many useful comments and ideas given while researching and preparing this paper.

References

[1] F. Chung, P. Diaconis, R. Graham, Universal cycles for combinatorial structures, Discrete Math. 110 (1992) 43–59.
[2] A. Lempel, On a homomorphism of the de Bruijn graph and its application to the design of feedback shift registers, IEEE Trans. Comput. C-19 (1970) 1204–1209.
[3] C.J. Mitchell, Constructing c-ary perfect factors, Des. Codes Cryptogr. 4 (1994) 341–368.
[4] C.J. Mitchell, T. Etzion, K.G. Paterson, A method for constructing decodable de Bruijn sequences, IEEE Trans. Inf. Theory 42 (1996) 1472–1478.
[5] K.G. Paterson, M.J.B. Robshaw, Storage efficient decoding for a class of binary de Bruijn sequences, Discrete Math. 138 (1995) 327–341.
[6] E.M. Petriu, J.S. Basran, On the position measurement of automated guided vehicles using pseudorandom encoding, IEEE Trans. Instrum. Meas. 38 (3) (1989) 799–803.
[7] J.R. Tuliani, On window sequences and position location Ph.D. Thesis, Royal Holloway and Bedford New College, University of London, 1998.