Efficient Integer Representations for Cryptographic Operations

by

James Alexander Muir

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2004

Abstract

Every positive integer has a unique radix 2 representation which uses the digits $\{0,1\}$. However, if we allow digits other than 0 and 1, say $\{0,1,-1\}$, then a positive integer has many representations. Of these *redundant* representations, it is possible to choose one that has few nonzero digits. It is well known that using representations of integers with few nonzero digits allows certain algebraic operations to be done more quickly. This thesis is concerned with various representations of integers that are related to efficient implementations of algebraic operations in cryptographic algorithms.

The topics covered here include:

–*The width-w nonadjacent form (w-NAF)*. We prove that the w-NAF of an integer has a minimal number of nonzero digits; that is, no other representation of an integer, which uses the w-NAF digits, can have fewer nonzero digits than its w-NAF.

–*A left-to-right analogue of the w-NAF*. We introduce a new family of radix 2 representations which use the same digits as the w-NAF, but have the property that they can be computed by sliding a window from left to right across the binary representation of an integer. We show these new representations have a minimal number of nonzero digits.

–*Joint representations*. Solinas introduced a $\{0,1,-1\}$-radix 2 representation for pairs of integers called the joint sparse form. We consider generalizations of the joint sparse form which represent $r \geq 2$ integers and use digits other than $\{0,1,-1\}$. We show how to construct a $\{0,1,2,3\}$-joint representation that has a minimal number of nonzero columns.

–*Nonadjacent digit sets*. It is well known that if x equals 3 or -1 then every nonnegative integer has a unique $\{0,1,x\}$-nonadjacent form; that is, a $\{0,1,x\}$-radix 2 representation with the property that, of any two consecutive digits, at most one is nonzero. We investigate what other values of x have this property.

Acknowledgements

My committee members, Guang Gong, Alfred Menezes, Jerry Solinas and Edlyn Teske, provided a number of helpful comments on this material and I thank them for patiently reading through my work.

I feel very fortunate to have had the opportunity to work with Doug Stinson during both my Masters and Ph.D. Doug has been an excellent supervisor and I have enjoyed developing a friendship with him over these years. I have also been fortunate to have benefited from the various activities of the Centre for Applied Cryptographic Research. Although many people contribute to the CACR, no one has invested more of their time and energy than Alfred Menezes. I thank Alfred for his efforts and for the time he has given me personally.

The Department of Combinatorics and Optimization provided a wonderful atmosphere during my graduate studies. It was a pleasure being part of a group where students, faculty and staff have a number of occasions to meet together and become better friends.

The first two years of my doctoral studies were partly supported by a Natural Science and Engineering Research Council postgraduate scholarship and I gratefully acknowledge their assistance.

I owe thanks to Bob and Virginia for welcoming me into their home and for providing many helpings of Sunday dinner. To my parents, Mary Jean and Jamie: thank-you for all your love and support. And finally, to Kristi:

If not for you
My sky would fall,
Rain would gather too.
Without your love I'd be nowhere at all,
Oh! What would I do
If not for you.

Bob Dylan, *If not for you*.

Contents

List of Figures

Chapter 1

Introduction

This thesis is concerned with various representations of integers that are related to efficient implementations of algebraic operations in cryptographic algorithms. As a concrete example, consider the Elliptic Curve Digital Signature Algorithm[1] (ECDSA). ECDSA is an elliptic curve based signature scheme that has been incorporated into a number of standards (e.g., FIPS 186-2 [10]). The ECDSA *signing* algorithm requires a computation of the form nP, and the *verification* algorithm requires a computation of the form $n_1P_1 + n_2P_2$; here, P, P_1, P_2 are points on an elliptic curve and n, n_1, n_2 are integers. We describe algorithms for performing these computations and then show how they can be made more efficient by using certain representations of the integers n, n_1, n_2.

The operation which takes an integer, n, and an elliptic curve point, P, and returns

$$nP := \underbrace{P + P + \cdots + P}_{n}$$

is called *scalar multiplication* or *point multiplication*. If we imagine a device (e.g., smartcard, pager) or a piece of software that computes nP, then one of the inputs accepted by this device/software would necessarily be the integer n. However, strictly speaking, the input would actually be some

[1]A good overview of ECDSA can be found in [14].

encoding of n. Exactly how n is encoded depends upon the application. A very common encoding is the binary number system, but there may be some other more convenient one. It might be that the device/software converts from one encoding to another.

Suppose that n is represented (i.e., encoded) as a string of digits, $a_{\ell-1} \dots a_1 a_0$, such that

$$n = a_{\ell-1}2^{\ell-1} + \cdots + a_2 2^2 + a_1 2^1 + a_0 2^0;$$

that is, $a_{\ell-1} \dots a_1 a_0$ is a *radix 2* representation of n. We denote sums like the one above by $(a_{\ell-1} \dots a_2 a_1 a_0)_2$.

If each a_i is in the set $\{0, 1\}$, then we can compute nP using the well known *binary method*:

```
$Q \leftarrow \infty$
**for** $i = \ell - 1 \dots 0$
  **do** $Q \leftarrow 2Q$
     **if** $a_i \neq 0$
       **then** $Q \leftarrow Q + P$
**return** $Q$
```

The operations $2Q$ and $Q + P$ are carried out by applying the elliptic curve group law which typically requires that a number of arithmetic operations be performed in the field over which the elliptic curve is defined. Computing $2Q$ and $Q + P$ are usually expensive operations. However, when Q is equal to the group identity element, ∞, then $2Q$ and $Q + P$ can be computed essentially for free.

The cost of the binary method is usually measured by counting the number of addition and doubling operations that are performed with $Q \neq \infty$. If the representation $(a_{\ell-1} \dots a_2 a_1 a_0)_2$ has $a_{\ell-1} \neq 0$ then the number of such doubling operations is $\ell - 1$ and the number of such addition operations is one less than the number of nonzero digits in $(a_{\ell-1} \dots a_2 a_1 a_0)_2$.

Example 1.1. For the integer 61, we have

$$61 = (111101)_2\,.$$

The number of addition operations required to compute $61P$ using this representation is 4 and the number of doubling operations is 5. $\diamondsuit$

Continuing with the example above, if we were to compute $61P$ by working through the binary method by hand, we would likely wish that the binary representation of 61 had fewer nonzero digits. However, every nonnegative integer has a unique binary representation so, unfortunately, we are stuck with 5 nonzero digits and 4 addition operations. But, the situation changes if we allow the digits a_i to take values other than 0 and 1.

Suppose each a_i is instead in the set $\{0, 1, -1\}$. Then we can compute nP like this:

```
$Q \leftarrow \infty$
for $i = \ell - 1 \ldots 0$
  do
    $Q \leftarrow 2Q$
    if $a_i \neq 0$
      then
        if $a_i > 0$
          then $Q \leftarrow Q + P$
          else $Q \leftarrow Q - P$
return $Q$
```

This method is sometimes called the *signed binary method*. In addition to the operations $2Q$ and $Q + P$, we now require $Q - P$. In an elliptic curve group, point *subtraction* can be done at essentially the same cost as point addition. A nonzero integer has many $\{0, \pm 1\}$-radix 2 representations (an infinite number, in fact) and *any* one of these can be used in the procedure above.

Example 1.2. Here are three $\{0, \pm 1\}$-radix 2 representations of 61:

$$61 = (1000\overline{1}01)_2 = (10\overline{1}\overline{1}101)_2 = (10000\overline{1}\overline{1})_2$$

We use $\overline{1}$ to denote the digit -1. Since each representation has the same length, when used to compute $61P$, all three result in 6 doubling operations. However, the first and third representations result in 2 addition/subtraction operations while the second results in 4.

To see that 61 has an infinite number of $\{0, \pm 1\}$-radix 2 representations, observe that we can always replace the digits 01 with $1\overline{1}$ and construct a new representation of 61. Thus,

$$\begin{aligned} 61 &= (1000\overline{1}01)_2 \\ &= (1\overline{1}000\overline{1}01)_2 \\ &= (1\overline{1}\overline{1}000\overline{1}01)_2 \\ &= (1\overline{1}\overline{1}\overline{1}000\overline{1}01)_2 \end{aligned}$$

and so on. $\Diamond$

This leads us to the following question: Given an integer n, what is the optimal $\{0, \pm 1\}$-radix 2 representation of n to use in the signed binary method to compute nP, with respect to minimizing the number of group operations? This problem can be expressed as follows:

$$\begin{aligned} &\min \ \mathsf{length}(\alpha) - 1 + \mathsf{wt}(\alpha) - 1 \\ &\quad \text{subject to} \\ &\quad \alpha \in \{0, \pm 1\}^*, \ n = (\alpha)_2. \end{aligned}$$

Here, the function $\mathsf{length}(\alpha)$ returns the number of digits that remain in the string α after we remove any leading zeros, and $\mathsf{wt}(\alpha)$ returns the number of nonzero digits in α.

A related minimization problem is the following:

$$\begin{aligned} &\min \ \mathsf{wt}(\alpha) \\ &\quad \text{subject to} \\ &\quad \alpha \in \{0, \pm 1\}^*, \ n = (\alpha)_2. \end{aligned}$$

If we take an optimal solution $\alpha = a_{\ell-1} \ldots a_1 a_0$ to this related problem and use it in the signed binary method, then nP will be computed using as few addition/subtraction operations as possible. In 1960, Rietweisner [39] explained how to construct an optimal solution and his results have been rediscovered many times (cf. [17]). He showed that every integer has a *unique* $\{0, \pm 1\}$-radix 2 representation with the property that, *of any two consecutive digits, at most one is nonzero*; moreover, he showed that no other $\{0, \pm 1\}$-radix 2 representation of n can have fewer nonzero digits than this canonical representation.

Rietweisner's canonical representations have come to be called *non-adjacent forms* (NAFs). In the previous example, it is easily checked that $(1000\bar{1}01)_2$ is the NAF of 61, while the other two representations have at least one pair of adjacent nonzero digits.

Taking $(a_{\ell-1} \ldots a_1 a_0)_2$ to be the NAF of n in the signed binary method minimizes the number of addition/subtraction operations. However, if the NAF of n was substantially longer than, say, the $\{0, 1\}$-radix 2 representation of n, then it would not be a good idea to use the NAF. Fortunately, the NAF of n is at most one digit longer than the $\{0, 1\}$-radix 2 representation of n.

We do not have to restrict ourselves to the digits $\{0, \pm 1\}$. In the general case, suppose we have $n = (a_{\ell-1} \ldots a_1 a_0)_2$ where each a_i is in a set of digits D, with $0 \in D$. If D has the property that $d \in D$ implies $|d| \in D$, then we can compute nP like this:

```
for each $d \in D$ with $d > 0$
  do $P_d \leftarrow dP$
$Q \leftarrow \infty$
for $i = \ell - 1 \ldots 0$
  do
    $Q \leftarrow 2Q$
    if $a_i \neq 0$
      then
        if $a_i > 0$
          then $Q \leftarrow Q + P_{a_i}$
          else $Q \leftarrow Q - P_{-a_i}$
return $Q$
```

This method differs from the previous ones in that it requires some *precomputation*; that is, the values P_d must be computed and stored before the representation $(a_{\ell-1} \ldots a_1 a_0)_2$ can be processed.[2] If we must compute nP for

[2]Actually, P_d only needs to be precomputed for those values of $d \in D$ with $d > 0$ such

several values of n then the values P_d need only be computed once since they can be stored and reused.

For certain families of digit sets, this procedure has been called a *window method*. The reason for this terminology is that for some digit sets a D-radix 2 representation of n can be constructed by applying *windows* of a certain width to the $\{0,1\}$-radix 2 representation of n.

Example 1.3. By applying a window of width 2 to the $\{0,1\}$-radix 2 representation of 61, we can construct a $\{0,1,3\}$-radix 2 representation as follows:

$$\begin{aligned} 61 &= (1111\boxed{01})_2 \\ &= (11\boxed{11}01)_2 \\ &= (\boxed{11}0301)_2 \\ &= (030301)_2. \end{aligned} \qquad \Diamond$$

Window methods allow nP to be computed using fewer group operations than the signed binary method, but this comes at the expense of using more memory.

If we use a window method to compute nP, then we can minimize the number of required addition/subtraction operations by constructing an optimal solution to the problem

$$\begin{aligned} \min\ & \mathrm{wt}(\alpha) \\ & \text{subject to} \\ & \alpha \in D^*,\ n = (\alpha)_2. \end{aligned}$$

Thus, for a fixed set of digits, it is of interest to investigate techniques for efficiently constructing an optimal solution to this problem. Similar questions and problems arise when we consider algorithms for computing the linear combination $n_1P_1 + n_2P_2$.

that d or $-d$ appears as a digit of $(a_{\ell-1}\ldots a_1a_0)_2$.

Suppose the integers n_1, n_2 are represented as strings of digits, $a_{\ell-1} \ldots a_1 a_0, b_{\ell-1} \ldots b_1 b_0$, such that

$$\begin{aligned} n_1 &= a_{\ell-1}2^{\ell-1} + \cdots + a_2 2^2 + a_1 2^1 + a_0 2^0, \\ n_2 &= b_{\ell-1}2^{\ell-1} + \cdots + b_2 2^2 + b_1 2^1 + b_0 2^0. \end{aligned}$$

Notice that these two strings have the same length. There is no loss of generality in imposing this condition since if one string was shorter than the other it could be padded on the left with zeros.

If each a_i and b_i is in the set $\{0, 1\}$, then we can compute $n_1 P_1 + n_2 P_2$ using the following method due to Straus [43]:

```
$Q \leftarrow \infty,\ R \leftarrow P_1 + P_2$
**for** $i = \ell - 1 \ldots 0$
**do**
    $Q \leftarrow 2Q$
    **if** $(a_i, b_i) \neq (0, 0)$
    **then**
        **if** $(a_i, b_i) = (1, 0)$
          **then** $Q \leftarrow Q + P_1$
        **else if** $(a_i, b_i) = (0, 1)$
          **then** $Q \leftarrow Q + P_2$
        **else if** $(a_i, b_i) = (1, 1)$
          **then** $Q \leftarrow Q + R$
**return** $Q$
```

As before, the cost of this method is determined by counting the number of doubling and addition operations. If one of $a_{\ell-1}, b_{\ell-1}$ is nonzero, then the number of doubling operations (with $Q \neq \infty$) is $\ell - 1$. To determine the number of addition operations, we need to examine the representations of n_1 and n_2 together. If we write the strings $a_{\ell-1} \ldots a_1 a_0$ and $b_{\ell-1} \ldots b_1 b_0$ in an array

$$\begin{array}{c} a_{\ell-1} \ldots a_1 a_0 \\ b_{\ell-1} \ldots b_1 b_0, \end{array}$$

then the number of addition operations (with $Q \neq \infty$) is equal to one less than the number of nonzero *columns* in this array.

We write
$$\begin{pmatrix} n_1 \\ n_2 \end{pmatrix} = \begin{pmatrix} a_{\ell-1} \dots a_2 a_1 a_0 \\ b_{\ell-1} \dots b_2 b_1 b_0 \end{pmatrix}_2$$
whenever $n_1 = (a_{\ell-1} \dots a_2 a_1 a_0)_2$ and $n_2 = (b_{\ell-1} \dots b_2 b_1 b_0)_2$. If $\mathcal{A}$ is a string composed of 2-digit columns, then $(\mathcal{A})_2$ is a radix 2 representation of a *pair* of integers. $(\mathcal{A})_2$ is called a *joint representation*.

Example 1.4. For the integers 602, 1365 we have
$$\begin{pmatrix} 602 \\ 1365 \end{pmatrix} = \begin{pmatrix} 01001011010 \\ 10101010101 \end{pmatrix}_2 .$$
This joint representation is composed of 11 columns, 9 of which are nonzero. Thus, the cost of computing $602P_1 + 1365P_2$ using Straus' method is 10 doublings and 8 additions.

Instead of using Straus' method, we could compute $602P_1 + 1365P_2$ by first computing $602P_1$ and $1365P_2$ using the binary method, and then adding the two results. This would cost $9 + 10 = 19$ doublings and $4 + 5 + 1 = 10$ additions. Thus, we see that Straus' method is less costly. $\lozenge$

Straus' method can be easily adapted to the case where each a_i and b_i is in the set $\{0, \pm 1\}$:

```
$Q \leftarrow \infty$, $R \leftarrow P_1 + P_2$, $S \leftarrow P_1 - P_2$
for $i = \ell - 1 \dots 0$
do
    $Q \leftarrow 2Q$
    if $(a_i, b_i) \neq (0,0)$
    then
        if $(a_i, b_i) = (1,0)$ then $Q \leftarrow Q + P_1$
        else if $(a_i, b_i) = (\bar{1},0)$ then $Q \leftarrow Q - P_1$
        else if $(a_i, b_i) = (0,1)$ then $Q \leftarrow Q + P_2$
        else if $(a_i, b_i) = (0,\bar{1})$ then $Q \leftarrow Q - P_2$
        else if $(a_i, b_i) = (1,1)$ then $Q \leftarrow Q + R$
        else if $(a_i, b_i) = (\bar{1},\bar{1})$ then $Q \leftarrow Q - R$
        else if $(a_i, b_i) = (1,\bar{1})$ then $Q \leftarrow Q + S$
        else if $(a_i, b_i) = (\bar{1},1)$ then $Q \leftarrow Q - S$
return $Q$
```

Now, we are faced with the same problem as before. A pair of nonzero integers has an infinite number of $\{0, \pm 1\}$-joint representations and anyone of these could be used to compute $n_1P_1 + n_2P_2$ in this procedure. The cost of computing $n_1P_1 + n_2P_2$ depends upon which $\{0, \pm 1\}$-joint representation we use. We would like to use a representation that minimizes the cost.

Example 1.5. Here are two $\{0, \pm 1\}$-joint representations of $(602, 1365)^T$:

$$\begin{pmatrix} 001010\bar{1}0\bar{1}010 \\ 10\bar{1}0\bar{1}0\bar{1}0\bar{1}0\overline{11} \end{pmatrix}_2, \begin{pmatrix} 01010\bar{1}0\bar{1}010 \\ 10101010101 \end{pmatrix}_2.$$

The first representation is composed of 12 columns, 7 of which are nonzero. The second representation has 11 columns and all 11 are nonzero. It is more efficient to use the first representation to compute $602P_1 + 1365P_2$. Notice that each row of the second representation is a NAF.

The problem of finding a $\{0, \pm 1\}$-joint representation that minimizes the number of addition/subtraction operations can be stated as

$$\begin{aligned} &\min \ \mathsf{wt}(\mathcal{A}) \\ &\qquad \text{subject to} \\ &\qquad \mathcal{A} \in (\{0, \pm 1\}^{2\times 1})^*,\ (n_1, n_2)^T = (\mathcal{A})_2. \end{aligned}$$

Here, $\mathsf{wt}(\mathcal{A})$ is the function that returns the number of nonzero columns in the representation $(\mathcal{A})_2$. As the previous example shows, taking each row of $(\mathcal{A})_2$ to be a NAF does not necessarily give an optimal solution.

This minimization problem was introduced by Solinas [42] and he has shown how to efficiently construct an optimal solution. Solinas' approach was to develop an analogue of the NAF for pairs of integers called the *joint sparse form* (JSF). A $\{0, \pm 1\}$-joint representation is a JSF if it satisfies the following properties:

1. Of any three consecutive columns, at most two are nonzero.

2. No two adjacent digits are $1\bar{1}$ or $\bar{1}1$.

3. For any two consecutive columns, if one row is 11 or $\overline{1}\overline{1}$, the other row is 10 or $\overline{1}0$.

It can be checked that the first representation in the previous example satisfies each of these properties. Every pair of integers has a *unique* JSF, and this representation is an optimal solution to the problem above.

Straus' method can also be adapted to other digit sets. Thus, for a given set of digits D and integers n_1, n_2, it is of interest to know if we can efficiently construct a D-joint representation of $(n_1, n_2)^T$ that has a minimal number of nonzero columns.

These examples hopefully demonstrate the kind of problems that we consider in this thesis.

1.1 Overview of Chapters

Chapter 2 We consider a generalization of the nonadjacent form called the *width-w nonadjacent form* (w-NAF). The w-NAF, where $w \geq 2$ is an integer, is a canonical radix 2 representation which uses the digits

$$D_w := \{0, \pm 1, \pm 3, \pm 5, \ldots, \pm(2^{w-1} - 1)\}.$$

Our main result is to prove that of all D_w-radix 2 representations of an integer, the w-NAF has a minimal number of nonzero digits. We also give a characterization of the w-NAF in terms of a colexicographical ordering. Lastly, we generalize a result on w-NAF and show that *any* D_w-radix 2 representation of an integer that has a minimal number of nonzero digits is at most one digit longer than its binary representation.

This material appears in the paper "Minimality and Other Properties of the Width-w Nonadjacent Form" [33] which will be published in *Mathematics of Computation*.

Chapter 3 Given the $\{0, 1\}$-radix 2 representation of an integer, its w-NAF can be constructed by sliding a window of width w from *right to left* across this representation. The w-NAF can then be used in a window method to efficiently compute a scalar multiple of an elliptic curve point. However, window methods generally work best by processing the digits of a representation from the opposite direction (i.e., from *left to right*). This means that the w-NAF must be computed and *stored* in its entirety before computations to determine the scalar multiple can begin. If the w-NAF digits could instead be calculated left-to-right, then storing the entire w-NAF representation would be unnecessary since its digits could be computed as they are needed.

We introduce a new family of radix 2 representations which use the same digits as the w-NAF but have the advantage that they result in a window method which uses less memory. This memory savings results from the fact that these new representations can be deduced using a very simple *left-to-right* algorithm. Further, we show that, like the w-NAF, these new representations have a minimal number of nonzero digits. We also give a characterization of these representations based on "closest approximations".

Some of this material appears in the paper "New Minimal Weight Representations for Left-to-Right Window Methods" [32] which will be presented at the Cryptographers' Track of the RSA Conference (February 2005), whose proceedings will be published in *Lecture Notes in Computer Science*.

Chapter 4 *Joint representations* can be defined as representations of *vectors* of integers, rather than just pairs of integers. As we have seen, radix 2 joint representations are useful for computing linear combinations of elliptic curve points (i.e., $\sum a_i P_i$). We consider the problem of constructing *minimal weight* r-row joint representations for various sets of digits.

Joint representations can be ordered *lexicographically* according to the position of their nonzero columns. We observe that minimal weight representations and lexicographically smallest representations share some of the same properties. Thus, for a given set of digits, it is natural to ask whether a lexicographically smallest representation has minimal weight. We show that, for certain families of digit sets, this is indeed true.

Chapter 5 The NAF of an integer is a radix 2 representation which uses the digits $\{0, 1, -1\}$ and has the property that, of any two consecutive digits, at most one is nonzero. Every positive integer has a unique NAF which can be efficiently computed. We investigate if other digit sets of the form $\{0, 1, x\}$ with $x \in \mathbb{Z}$ provide each positive integer with a "nonadjacent" representation; that is, a representation with the property that, of any two consecutive digits, at most one is nonzero. If a digit set has this property, we call it a *nonadjacent digit set* (NADS).

We present an algorithm to determine if $\{0, 1, x\}$ is a NADS; and if it is, we present an algorithm to efficiently determine the nonadjacent representation of any positive integer. We also present some necessary and sufficient conditions for $\{0, 1, x\}$ to be a NADS. These conditions are used to exhibit infinite families of integers x such that $\{0, 1, x\}$ is a NADS, as well as infinite families of x such that $\{0, 1, x\}$ is not a NADS.

This material appears in the paper "Alternative Digit Sets for Nonadjacent Representations" [31] which will be published in *SIAM Journal on Discrete Mathematics*. A preliminary version of this paper [30] was presented at the Workshop on Selected Areas in Cryptography in August 2003 with the proceedings published in volume 3006 of *Lecture Notes in Computer Science*.

1.2 Preliminaries

Most of the notation and definitions we use will be introduced as required. However, there are some concepts that we rely on quite frequently and so we make a note of them here.

Definition 1.6. A *radix 2 representation* is a *finite* sum of the form $\sum_{i\geq 0} a_i 2^i$.

If n is an integer and $n = \sum_{i\geq 0} a_i 2^i$, we call $\sum_{i\geq 0} a_i 2^i$ a *radix 2 representation* of n. To denote radix 2 representations, the following notation is commonly used:

$$(\dots a_3 a_2 a_1 a_0)_2 = \cdots + a_3 2^3 + a_2 2^2 + a_1 2^1 + a_0.$$

Each of the a_i's is called a *digit*. The *binary number system* is the radix 2 number system where each digit is equal to either 0 or 1.

Since $(\dots a_2 a_1 a_0)_2$ stands for a finite sum, all but a finite number of the a_i's are zero. Because of this property, we can consider the *length* of a representation:

Definition 1.7. The *length* of a representation $(\dots a_2 a_1 a_0)_2$ is the integer

$$\min\{\ell \in \mathbb{Z} : \ell \geq 0, \text{and for any } i \geq \ell,\ a_i = 0\}.$$

For the representation $(a_{\ell-1} \dots a_1 a_0)_2$, it is implicit that $a_i = 0$ for all $i \geq \ell$; note that if $a_{\ell-1} \neq 0$, then this representation has length ℓ.

If $D \subset \mathbb{Z}$ is a set of digits, the set of all *strings* of digits from D is denoted by D^*. The empty string is in D^* and is denoted by ϵ.

Now, given a representation $(a_{\ell-1} \dots a_1 a_0)_2$, where each a_i is in D, then $a_{\ell-1} \dots a_1 a_0$ is a string in D^*. Conversely, any string $\alpha \in D^*$ corresponds to a radix 2 representation with digits in D, namely $(\alpha)_2$. If $\alpha, \beta \in D^*$ then we denote their *concatenation* by $\alpha \| \beta$. Also, we denote the string formed by deleting the leading zeros from α by $\widehat{\alpha}$.

It is important to note that representations and strings have different properties. For example, the strings $300\overline{3}0$ and $00300\overline{3}0$ are different, however, $(300\overline{3}0)_2$ and $(00300\overline{3}0)_2$ denote the same representation.

If α is a string of digits then we write $\mathsf{wt}(\alpha)$ to denote the number of nonzero digits in α. The value $\mathsf{wt}(\alpha)$ is often called the *Hamming weight* of α.

Chapter 2

The Width-w Nonadjacent Form

2.1 Introduction

Let $w \geq 2$ be an integer. A radix 2 representation is called a *width-w nonadjacent form* (w-NAF, for short) if it satisfies the following conditions:

1. Each nonzero digit is an odd integer with absolute value less than 2^{w-1}.

2. Of any w consecutive digits, at most one is nonzero.

It is convenient to define D_w to be the set of w-NAF digits; that is, D_w is the set of integers which includes zero and the odd integers with absolute value less than 2^{w-1}. For example, if $w = 3$ then $D_w = \{0, \pm 1, \pm 3\}$. The number 42 has a 3-NAF since the representation $(300\bar{3}0)_2$ (note that $\bar{1}$ denotes -1, $\bar{3}$ denotes -3, etc.) satisfies conditions (1) and (2), and

$$(300\bar{3}0)_2 = 3 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 - 3 \cdot 2^1 + 0 \cdot 2^0 = 42.$$

When $w = 2$, $D_w = \{0, \pm 1\}$ and the w-NAF coincides with the well known *nonadjacent form* [11]. Because of this, the w-NAF may be regarded as a generalization of the ordinary NAF.

Cryptographers became interested in the w-NAF primarily through efforts to efficiently implement elliptic curve scalar multiplication. It was known that, on average, the w-NAF of an integer has few nonzero digits, however, except in the case when $w = 2$, it was not known if the w-NAF has a minimal number of nonzero digits,

We provide an answer to this question in Section 2.3 of this chapter: we prove that no other D_w-radix 2 representation of an integer has fewer nonzero digits than its w-NAF. This result complements the average case analysis carried out in [8] and provides further evidence that the w-NAF is a good representation to use to perform scalar multiplication. As well, this result may also have applications to the theory of arithmetic codes [23].

In Section 2.4, we generalize a known result about the length of w-NAFs. It is stated without proof in [28] that the length of the w-NAF of an integer is *at most one digit longer* than its binary representation. We show that this is in fact a property of representations with a minimal number of nonzero digits; that is, any D_w-radix 2 representation of an integer with a minimal number of nonzero digits is at most one digit longer than its binary representation.

In Section 2.5, we provide a new characterization of the w-NAF in terms of a *colexicographical ordering*. For an integer n, we consider the set of all D_w-radix 2 representations of n. The position of the zero and nonzero digits in these representations define binary strings. When we order these strings colexicographically, we show that there is always a unique smallest string; moreover, the representation to which this unique smallest string corresponds is the w-NAF of n.

Before we present our results, we first establish some of the basic theory on w-NAFs in Section 2.2. Aside from being of value to readers new to the w-NAF, this material provides proofs for some results which are stated without proof in the literature.

2.2 Known Results

The w-NAF seems to have been first described by Cohen, Miyaji and Ono [9]. However, the w-NAF is closely related to the binary window method and this may explain why it was proposed independently by Blake, Seroussi and Smart [5] and by Solinas [41].

Results on the w-NAF are scattered among different papers and often proofs are not given. For completeness, we give proofs of the following basic facts about the w-NAF:

1. every integer has at most one w-NAF.
2. every integer has a w-NAF.
3. an integer's w-NAF is at most one digit longer than its binary representation.

2.2.1 Uniqueness

Proposition 2.1. *Every integer has at most one w-NAF.*

Proof. We suppose the result is false and show that this leads to a contradiction. Suppose there are two different w-NAFs, say $(a_{\ell-1} \dots a_2 a_1 a_0)_2$ and $(b_{\ell'-1} \dots b_2 b_1 b_0)_2$, such that

$$(a_{\ell-1} \dots a_2 a_1 a_0)_2 = (b_{\ell'-1} \dots b_2 b_1 b_0)_2,$$

where ℓ and ℓ' are the respective lengths of these representations. We can assume that ℓ is as small as possible. These representations stand for the same integer, call it n.

If $a_0 = b_0$, then

$$(a_{\ell-1} \dots a_2 a_1)_2 = (b_{\ell'-1} \dots b_2 b_1)_2,$$

and so we have two different, and shorter, w-NAFs which stand for the same integer, contrary to the minimality of ℓ. So, it must be that $a_0 \neq b_0$.

If n is even then $a_0 = b_0 = 0$. However, $a_0 \neq b_0$, so it must be that n is odd; hence, both a_0 and b_0 are nonzero. Because the representations are both w-NAFs, we have

$$\begin{aligned}(a_{\ell-1} \ldots a_w 00 \ldots 0 a_0)_2 &= (b_{\ell'-1} \ldots b_w 00 \ldots 0 b_0)_2 \\ \implies a_0 &\equiv b_0 \pmod{2^w}.\end{aligned}$$

However, $-(2^{w-1}-1) \leq a_0, b_0 \leq 2^{w-1}-1$, and thus

$$-2(2^{w-1}-1) \leq a_0 - b_0 \leq 2(2^{w-1}-1).$$

The only multiple of 2^w in this range is 0, and since $2^w | (a_0 - b_0)$ it must be that $a_0 - b_0 = 0$. However, this contradicts the fact that $a_0 \neq b_0$. Thus, the representations cannot exist and the result follows. □

2.2.2 Existence

We present an algorithm which, on input n, computes a string α such that $(\alpha)_2$ is a w-NAF of n. Unlike the algorithms in [5] and [41], our algorithm handles negative integers as well as positive ones. Proving that the algorithm is correct establishes that every integer has a w-NAF.

The quotient-remainder theorem tells us that, for any integer n, there exist unique integers q' and r' such that

$$n = q' \cdot 2^w + r' \quad \text{where} \quad 0 \leq r' < 2^w.$$

It is common to denote this value of r' by "$n \bmod 2^w$". It follows that there also exist unique integers q and r such that

$$n = q \cdot 2^w + r \quad \text{where} \quad -2^{w-1} < r \leq 2^{w-1}.$$

We will denote this value of r by "$n \text{ mods } 2^w$". For example, if $w = 3$ then $13 \text{ mods } 2^3 = -3$. Note that if n is odd then so is $n \text{ mods } 2^w$. As well, when $n > 0$ it must be that $q \geq 0$, and similarly, when $n < 0$, $q \leq 0$. So, for $n \neq 0$, we have $q/n \geq 0$.

Our algorithm makes use of the following two functions:

$$f_w(n) := \begin{cases} n/2 & \text{if } n \text{ is even,} \\ (n-r)/2^w & \text{where } r = n \text{ mods } 2^w \text{, otherwise.} \end{cases} \tag{2.1}$$

$$g_w(n) := \begin{cases} 0 & \text{if } n \text{ is even,} \\ 0^{w-1}r & \text{where } r = n \text{ mods } 2^w \text{, otherwise.} \end{cases} \tag{2.2}$$

Note that f_w returns an integer and g_w returns a string. For example, if $w = 3$, then $f_3(13) = 2$ and $g_3(13) = 00\overline{3}$.

Now we can describe our algorithm:

```
**Algorithm 2.2:** NAF_w(n)

$\alpha \leftarrow \epsilon$
**while** $n \neq 0$
    **do** $\begin{cases} \alpha \leftarrow g_w(n) \parallel \alpha \\ n \leftarrow f_w(n) \end{cases}$
**return** $\widehat{\alpha}$
```

As $\text{NAF}_w(n)$ executes, it builds a string, α, in ${D_w}^*$. And assuming $\text{NAF}_w(n)$ terminates, which we will prove in a moment, it returns this string minus its leading zeros (i.e., $\widehat{\alpha}$).

We justify the title "Algorithm" by showing that $\text{NAF}_w(n)$ terminates for all $n \in \mathbb{Z}$. If $n = 0$, then $\text{NAF}_w(n)$ clearly terminates, so we need only consider $n \neq 0$. We will argue that $|f_w(n)| < |n|$ whenever $n \neq 0$.

If n is even, then $f_w(n) = n/2$ and thus $|f_w(n)| < |n|$. If n is odd, then we consider two cases. First, suppose $|n| < 2^{w-1}$. Then we see that $n \text{ mods } 2^w = n$ and thus

$$|f_w(n)| = 0 < |n|.$$

Second, suppose $|n| \geq 2^{w-1}$; then we have

$$|f_w(n)| = \left|\frac{n-r}{2^w}\right| \leq \left|\frac{n}{2^w}\right| + \left|\frac{r}{2^w}\right| < \left|\frac{n}{2^w}\right| + \frac{1}{2}.$$

Now,

$$\frac{1}{2} = \frac{2^{w-1}}{2^w} \le \frac{|n|}{2^w},$$

and, since $w \ge 2$, it follows that

$$|f_w(n)| < 2\left|\frac{n}{2^w}\right| = \left|\frac{n}{2^{w-1}}\right| < |n|\,.$$

So, the sequence formed by taking the absolute value of the variable n during the execution of $\mathrm{NAF}_w(n)$ is strictly decreasing. Thus, the variable n must reach 0, and so $\mathrm{NAF}_w(n)$ terminates for all $n \in \mathbb{Z}$.

Now we argue that the algorithm is correct.

Proposition 2.3. *Let α be the string returned by $NAF_w(n)$ where $n \in \mathbb{Z}$. Then $(\alpha)_2$ is a w-NAF and $(\alpha)_2 = n$.*

Proof. By the definition of g_w, it is clear that $(\alpha)_2$ is a w-NAF, so we just have to show that $(\alpha)_2 = n$. If i is a nonnegative integer, we write ${f_w}^i$ to denote i applications of the map f_w; that is,

$${f_w}^i = \underbrace{f_w \circ f_w \circ \cdots \circ f_w}_{i}\,.$$

Since $\mathrm{NAF}_w(n)$ terminates, there is some integer $i \ge 0$ such that ${f_w}^i(n) = 0$. We will argue by induction on i.

When $i = 0$, ${f_w}^i(n) = 0$ implies $n = 0$. For $n = 0$, we have $\alpha = \epsilon$ and then $n = (\alpha)_2$ as required. Suppose $i > 0$. Let $n' = f_w(n)$ and let α' be the string returned by $\mathrm{NAF}_w(n')$. Note that ${f_w}^{i-1}(n') = {f_w}^i(n) = 0$ and so by induction we have that $n' = (\alpha')_2$. By the definition of Algorithm 2.2 we see that

$$\begin{aligned}
&\alpha = \alpha' \| g_w(n) \\
\Longrightarrow\ &(\alpha)_2 = (\alpha' \| g_w(n))_2 \\
\Longrightarrow\ &(\alpha)_2 = 2^{|g_w(n)|}(\alpha')_2 + (g_w(n))_2 \\
\Longrightarrow\ &(\alpha)_2 = 2^{|g_w(n)|} n' + (g_w(n))_2 \\
\Longrightarrow\ &(\alpha)_2 = 2^{|g_w(n)|} f_w(n) + (g_w(n))_2 \qquad (2.3)
\end{aligned}$$

From (2.1), we see the function f_w can be defined in terms of g_w as follows:

$$f_w(n) = \frac{n - (g_w(n))_2}{2^{|g_w(n)|}}.$$

Thus, the right-hand side of (2.3) equals n, and so $(\alpha)_2 = n$ as required. □

Because of Propositions 2.1 and 2.3, we now know that each integer n has a unique w-NAF. Henceforth, we will refer to this representation as *the* w-NAF of n.

2.2.3 Length

We show that the length of the w-NAF of n is at most one digit longer than the $\{0, 1\}$-radix 2 representation of $|n|$. This fact seems to have been first stated, without proof, by Möller [28]. A more general result is proved in Section 4; however, we feel it is of interest to provide a direct proof here.

We start with a Lemma.

Lemma 2.4. *Let $(a_{\ell-1} \ldots a_1 a_0)_2$ be a w-NAF of length ℓ where $\ell \geq 1$. If $n = (a_{\ell-1} \ldots a_1 a_0)_2$ then $n > 0$ if and only if $a_{\ell-1} > 0$.*

Proof. Note that since the length of $(a_{\ell-1} \ldots a_1 a_0)_2$ is ℓ we have $a_{\ell-1} \neq 0$. We argue by induction on ℓ. The result is clearly true when $\ell = 1$. Suppose $\ell > 1$.

If $a_0 = 0$ we let $n' = (a_{\ell-1} \ldots a_1)_2$. Then we see that

$$n > 0 \iff 2n' > 0 \iff n' > 0 \iff a_{\ell-1} > 0,$$

where the last equivalence follows by induction.

If $a_0 \neq 0$, then

$$a_{\ell-1} \ldots a_w \ldots a_1 a_0 = a_{\ell-1} \ldots a_w 0 \ldots 0 a_0,$$

since $(a_{\ell-1} \ldots a_1 a_0)_2$ is a w-NAF. Thus,

$$n = 2^w (a_{\ell-1} \ldots a_w)_2 + a_0 \quad \text{where} \quad -2^{w-1} < a_0 \leq 2^{w-1}.$$

Since $a_{\ell-1} \neq 0$, we have $(a_{\ell-1} \dots a_w)_2 \neq 0$, thus

$$n > 0 \iff (a_{\ell-1} \dots a_w)_2 > 0 \iff a_{\ell-1} > 0,$$

where the last equivalence follows by induction. This proves the result. $\square$

Proposition 2.5. *For any integers n, w, where $w \geq 2$, the length of the w-NAF of n is at most one digit longer than the binary representation of $|n|$.*

Proof. Let ℓ be the length of the w-NAF of $|n|$ and let m be the length of the binary representation of $|n|$. If $n = 0$, then $\ell = m = 0$, and so the result is true. Suppose $n \neq 0$. Let $(a_{\ell-1} \dots a_1 a_0)_2$ be the w-NAF of $|n|$. Since $|n|$ is positive, by Lemma 2.4 we have that $a_{\ell-1} \geq 1$. Let $a = -(2^{w-1} - 1)$. Note that,

$$\begin{aligned}
& (a_{\ell-1} a_{\ell-2} \dots a_1 a_0)_2 = |n| \\
\implies & (1 a_{\ell-2} \dots a_1 a_0)_2 \leq |n| \\
\implies & (1 \underbrace{00\dots0a}_{w} \underbrace{00\dots0a}_{w} \dots)_2 \leq |n| \\
\implies & (\underbrace{11\dots1}_{w} \underbrace{aa\dots a}_{w} \underbrace{aa\dots a}_{w} \dots)_2 \leq (2^{w-1} + \dots + 2^2 + 2^1 + 1)\,|n|\,.
\end{aligned}$$

Consider the representation on the left-hand side of this last inequality. Reading from left to right, its digits consist of a run of ones, followed by a run of a's, ended by a (possibly empty) run of zeros. If we replace this run of zeros with a run of a's, then we have

$$\begin{aligned}
& (\underbrace{11\dots1}_{w} \underbrace{aa\dots a}_{\ell-1})_2 \leq (2^{w-1} + \dots + 2^2 + 2^1 + 1)\,|n| \\
\implies & 2^{\ell-1}(2^w - 1) + a(2^{\ell-1} - 1) \leq (2^w - 1)\,|n| \\
\implies & 2^{\ell-1}(2^w - 1) - (2^{w-1} - 1)(2^{\ell-1} - 1) \leq (2^w - 1)\,|n| \\
\implies & 2^{\ell-1} - \frac{2^{w-1} - 1}{2^w - 1}(2^{\ell-1} - 1) \leq |n| \\
\implies & 2^{\ell-1} - \frac{1}{2}(2^{\ell-1} - 1) < |n| \\
\implies & 2^{\ell-2} + \frac{1}{2} < |n| \\
\implies & 2^{\ell-2} < |n|\,.
\end{aligned}$$

Now, from the binary representation of $|n|$, we have $|n| < 2^m$, thus

$$\begin{aligned} 2^{\ell-2} &< 2^m \\ \implies \ell - 2 &< m \\ \implies \ell - m &\leq 1. \end{aligned}$$

This gives us the required result. □

2.3 Minimality

The main topic of this section is to prove that the w-NAF has a minimal number of nonzero digits; that is, we want to show that no other representation of an integer, with digits in D_w, has fewer nonzero digits than its w-NAF. We begin with a discussion of addition of representations. We will see that the properties of addition provide a key step in our proof of minimality.

For any $\alpha \in {D_w}^*$ and $c_0 \in \mathbb{Z}$ with $|c_0| < 2^{w-1}$, we show that there exists some $\beta \in {D_w}^*$ such that $(\beta)_2 = (\alpha)_2 + c_0$ and

$$\mathsf{wt}(\beta) \leq \mathsf{wt}(\alpha) + 1. \tag{2.4}$$

We do so by developing a certain algorithm for addition.

Given α and c_0, we want to compute a representation $\beta \in {D_w}^*$ with $(\beta)_2 = (\alpha)_2 + c_0$. Let $\alpha = \ldots a_2 a_1 a_0$ and $\beta = \ldots b_2 b_1 b_0$. To compute the sum we define a sequence of integers $c_1, c_2, \ldots$. Writing our variables in the following array suggests how these values are related:

$$\begin{array}{r} \ldots \overset{c_3}{a_3}\overset{c_2}{a_2}\overset{c_1}{a_1}a_0 \\ +\underline{\qquad\quad c_0} \\ \ldots b_3 b_2 b_1 b_0 \end{array}$$

Starting with $i = 0$, we examine a_i and c_i and then assign values to b_i and

c_{i+1}. The following rules are used to define b_i:

$a_i \bmod 2$	$c_i \bmod 2$	b_i
0	0	a_i
0	1	c_i
1	0	a_i
1	1	0

Further, c_{i+1} is always set to the value $(a_i + c_i - b_i)/2$.

We claim the representation β is in ${D_w}^*$. To justify this claim, we first show that each c_{i+1} satisfies $|c_{i+1}| < 2^{w-1}$. Note that $b_i \in \{0, a_i, c_i\}$. Since $a_i \in D_w$, we have $|a_i| < 2^{w-1}$, and by induction $|c_i| < 2^{w-1}$. Thus,

$$\begin{aligned} b_i = 0 &\implies c_{i+1} = \frac{a_i + c_i}{2} \implies |c_{i+1}| < 2^{w-1} \\ b_i = a_i &\implies c_{i+1} = \frac{c_i}{2} \implies |c_{i+1}| < 2^{w-2} \\ b_i = c_i &\implies c_{i+1} = \frac{a_i}{2} \implies |c_{i+1}| < 2^{w-2}. \end{aligned}$$

Now, it is easy to see that $\beta \in {D_w}^*$. If b_i equals 0 or a_i, then clearly $b_i \in D_w$. If $b_i = c_i$, then, according to our rules, it must be that c_i is odd. Since c_i is odd and $|c_i| < 2^{w-1}$, $c_i \in D_w$. Hence, $b_i \in D_w$ for all i.

Based on the grade-school method of addition, it may seem natural for the rules which define b_i to be implemented inside an appropriate "for" loop. However, for our purposes, it is more convenient if we take a different approach. We first initialize the string $\beta = \ldots b_2 b_1 b_0$ to equal $\alpha = \ldots a_2 a_1 a_0$ and then correct the digits of β as necessary. Here is a description of this process in pseudocode:

Algorithm 2.6: ADD-DIGIT(α, c_0)

```
comment: $\alpha = \ldots a_2 a_1 a_0,\ a_i \in D_w,\ |c_0| < 2^{w-1}$
$\ldots b_2 b_1 b_0 \leftarrow \ldots a_2 a_1 a_0$
$i \leftarrow 0$
while $c_i \neq 0$
  do
    $(a, c) \leftarrow (a_i, c_i) \mod 2$
    if $(a, c) = (0, 1)$
      then $b_i \leftarrow c_i$
    else if $(a, c) = (1, 1)$
      then $b_i \leftarrow 0$
    $c_{i+1} \leftarrow (a_i + c_i - b_i)/2$
    $i \leftarrow i + 1$
return $\beta = \ldots b_2 b_1 b_0$
```

For the input $\alpha = \ldots a_2 a_1 a_0$, let $\ell - 1$ be the largest value of i such that $a_i \neq 0$ (thus, the representation $(\alpha)_2$ has length ℓ). By convention, we let $a_i = 0$ for all $i \geq \ell$. The algorithm terminates if and only if the sequence $c_0, c_1, c_2, \ldots$ reaches zero. If none of $c_0, c_1, \ldots, c_\ell$ are equal to zero then certainly one of $c_{\ell+1}, c_{\ell+2}, \ldots$ will be; this is because for $i \geq \ell$, c_{i+1} is equal to either 0 or $c_i/2$. Thus, we see that ADD-DIGIT(α, c_0) always terminates.

A short example helps illustrate how the algorithm works. Let $w = 4$, then $D_4 = \{0, \pm 1, \pm 3, \pm 5, \pm 7\}$. Suppose $\alpha = 13570001357$ and $c_0 = 6$. Then the algorithm adds $(13570001357)_2$ and 6 as follows:

$$\begin{array}{r} {\scriptstyle 0\,1\,2\,4\,3} \\ 13570001357 \\ +\underline{6} \\ 13570011307 \end{array}$$

It is interesting to note that ADD-DIGIT computes a sum in ${D_w}^*$ without using the operator "mods 2^w".

The following Lemma verifies that the algorithm is correct.

Lemma 2.7. *Let β be the string returned by* ADD-DIGIT(α, c_0) *where $\alpha \in {D_w}^*$ and $|c_0| < 2^{w-1}$. Then, $(\beta)_2 = (\alpha)_2 + c_0$.*

Proof. Let i^* be the value of i when ADD-DIGIT(α, c_0) returns. We argue by induction on i^*. If $i^* = 0$ then

$$\begin{aligned} i^* = 0 &\implies c_0 = 0 \text{ and } \beta = \alpha \\ &\implies (\beta)_2 = (\alpha)_2 + c_0. \end{aligned}$$

So, the result holds for $i^* = 0$.

Suppose $i^* > 0$. From the strings $\alpha = \ldots a_2 a_1 a_0$ and $\beta = \ldots b_2 b_1 b_0$, we define $\alpha' = \ldots a_2 a_1$ and $\beta' = \ldots b_2 b_1$. Let $c_1, c_2, \ldots$ be the sequence of carries which occurs during the computation of ADD-DIGIT(α, c_0). From the description of Algorithm 2.6, we see that ADD-DIGIT(α', c_1) must return the string β'. Moreover, the value of i when ADD-DIGIT(α', c_1) returns must be $i^* - 1$. Now,

$$\begin{aligned} & (\beta')_2 = (\alpha')_2 + c_1 \qquad \text{(by induction)} \\ \implies & (\beta' \| 0)_2 = (\alpha' \| 0)_2 + 2c_1 \\ \implies & (\beta' \| 0)_2 + b_0 = (\alpha' \| 0)_2 + 2c_1 + b_0 \\ \implies & (\beta)_2 = (\alpha' \| 0)_2 + a_0 + c_0 \qquad \text{(since } c_1 = (a_0 + c_0 - b_0)/2) \\ \implies & (\beta)_2 = (\alpha)_2 + c_0. \end{aligned}$$

□

Returning to (2.4), if we are given $\alpha \in {D_w}^*$ and c_0, with $|c_0| < 2^{w-1}$, we can use ADD-DIGIT(α, c_0) to compute a string $\beta \in {D_w}^*$ such that $(\beta)_2 = (\alpha)_2 + c_0$. We will show that $\mathsf{wt}(\beta) \le \mathsf{wt}(\alpha) + 1$.

Lemma 2.8. *Let β be the string returned by* ADD-DIGIT(α, c_0) *where $\alpha \in {D_w}^*$ and $|c_0| < 2^{w-1}$. Then, $\mathsf{wt}(\beta) \le \mathsf{wt}(\alpha) + 1$.*

Proof. Let i^* be the value of i when ADD-DIGIT(α, c_0) returns. Consider the sequence $t_0, t_1, \ldots, t_{i^*}$ where

$$t_i = \mathsf{wt}(\ldots a_{i+1} a_i) + \mathsf{wt}(c_i) + \mathsf{wt}(b_{i-1} \ldots b_1 b_0)\,.$$

We argue that this sequence is monotonically decreasing. Once we establish this fact, we show that the Lemma follows.

First, note that from the description of Algorithm 2.6, we have

$$c_{i+1} = \frac{a_i + c_i - b_i}{2}, \quad \text{for} \quad 0 \leq i \leq i^* - 1.$$

Now, $b_i \in \{0, a_i, c_i\}$, and after applying a simple case analysis to this equality we can conclude that

$$\mathsf{wt}(c_{i+1}) \leq \mathsf{wt}(a_i) + \mathsf{wt}(c_i) - \mathsf{wt}(b_i)\,. \tag{2.5}$$

For example, suppose $b_i = 0$. Then we must show that $\mathsf{wt}(c_{i+1}) \leq \mathsf{wt}(a_i) + \mathsf{wt}(c_i)$. Since $i < i^*$, we have $c_i \neq 0$ and thus $\mathsf{wt}(c_i) = 1$. Now,

$$\mathsf{wt}(c_{i+1}) \leq 1 = \mathsf{wt}(c_i) \leq \mathsf{wt}(a_i) + \mathsf{wt}(c_i)\,.$$

The other cases are argued in a similar manner.

For $0 \leq i \leq i^* - 1$ we need to show that $t_i \geq t_{i+1}$. If we compare

$$t_i = \mathsf{wt}(\ldots a_{i+1}a_i) + \mathsf{wt}(c_i) + \mathsf{wt}(b_{i-1}\ldots b_1 b_0)$$

to

$$t_{i+1} = \mathsf{wt}(\ldots a_{i+2}a_{i+1}) + \mathsf{wt}(c_{i+1}) + \mathsf{wt}(b_i \ldots b_1 b_0)$$

and eliminate equal digits, we see that

$$t_i \geq t_{i+1} \iff \mathsf{wt}(a_i) + \mathsf{wt}(c_i) \geq \mathsf{wt}(c_{i+1}) + \mathsf{wt}(b_i)\,.$$

However, this last inequality holds by (2.5). Thus, the sequence of t_i's is monotonically decreasing.

Since $t_0 \geq t_1 \geq \cdots \geq t_{i^*}$ we have $t_0 \geq t_{i^*}$. Note that

$$t_0 = \mathsf{wt}(\ldots a_1 a_0) + \mathsf{wt}(c_0) = \mathsf{wt}(\alpha) + \mathsf{wt}(c_0)\,.$$

Since $b_i = a_i$, for $i \geq i^*$, and $c_{i^*} = 0$ we have

$$t_{i^*} = \mathsf{wt}(\ldots a_{i^*+1}a_{i^*}) + \mathsf{wt}(c_{i^*}) + \mathsf{wt}(b_{i^*-1}\ldots b_1 b_0) = \mathsf{wt}(\beta)\,.$$

Thus, from $t_{i^*} \leq t_0$, we can conclude that

$$\mathsf{wt}(\beta) \leq \mathsf{wt}(\alpha) + \mathsf{wt}(c_0) \leq \mathsf{wt}(\alpha) + 1.$$

□

Now we have all the tools we need to proceed with our main result.

Theorem 2.9. *If* $(\alpha)_2$ *is a* w*-NAF then for any* $\beta \in D_w{}^*$ *with* $(\beta)_2 = (\alpha)_2$*, we have* $\mathsf{wt}(\alpha) \leq \mathsf{wt}(\beta)$.

Proof. Suppose the result is false. Then for some w-NAF, $(\alpha)_2$, there exists $\beta \in D_w{}^*$ with $(\beta)_2 = (\alpha)_2$ and $\mathsf{wt}(\alpha) > \mathsf{wt}(\beta)$. Choose $(\alpha)_2$ so that its length is minimal. Any w-NAF with length less than that of $(\alpha)_2$ must have a minimal number of nonzero digits.

Let $\alpha = \ldots a_2 a_1 a_0$ and $\beta = \ldots b_2 b_1 b_0$. If $a_0 = b_0$ then $(\ldots a_2 a_1)_2 = (\ldots b_2 b_1)_2$ and so $(\ldots a_2 a_1)_2$ is a shorter counter-example. However, this contradicts our choice of $(\alpha)_2$, so it must be that $a_0 \neq b_0$. Consequently, $(\alpha)_2$ must be odd, since otherwise $a_0 = b_0 = 0$. Hence, both a_0 and b_0 are nonzero.

Since $(\alpha)_2$ is a w-NAF we have

$$\alpha = \ldots a_w 00 \ldots 0 a_0.$$

Write

$$\alpha = \alpha_1 \| \overbrace{00 \ldots 0 a_0}^{w} \qquad \text{and} \qquad \beta = \beta_1 \| \overbrace{b_{w-1} \ldots b_1 b_0}^{w}$$

where $\alpha_1, \beta_1 \in D_w{}^*$. Note that since $(\alpha)_2$ is a w-NAF, so is $(\alpha_1)_2$, and further, $\mathsf{wt}(\alpha) = \mathsf{wt}(\alpha_1) + 1$.

We show that at least two of the digits in the string $b_{w-1} \ldots b_1 b_0$ must be nonzero. Suppose not; then all of the digits $b_{w-1} \ldots b_1 b_0$ are zero except for b_0, and so

$$\begin{aligned} (\alpha)_2 &= (\beta)_2 \\ \implies (\alpha_1 \| 00 \ldots 0 a_0)_2 &= (\beta_1 \| 00 \ldots 0 b_0)_2 \\ \implies a_0 &\equiv b_0 \pmod{2^w} \\ \implies a_0 &= b_0 \qquad (\text{since } a_0, b_0 \in D_w). \end{aligned}$$

But this is a contradiction since a_0 and b_0 cannot be equal. So $\mathsf{wt}(b_{w-1} \ldots b_1 b_0) \geq 2$, and hence $\mathsf{wt}(\beta) \geq \mathsf{wt}(\beta_1) + 2$.

Now,

$$\begin{aligned}(\alpha)_2 &= (\beta)_2\\ \implies (\alpha_1\|00\ldots 0a_0)_2 &= (\beta_1\|b_{w-1}\ldots b_1b_0)_2\\ \implies (\alpha_1)_2\cdot 2^w + (00\ldots 0a_0)_2 &= (\beta_1)_2\cdot 2^w + (b_{w-1}\ldots b_1b_0)_2\\ \implies (\alpha_1)_2 &= (\beta_1)_2 + \frac{(b_{w-1}\ldots b_1b_0)_2 - (00\ldots 0a_0)_2}{2^w}.\end{aligned}$$

Let $c_0 = \left((b_{w-1}\ldots b_1b_0)_2 - (00\ldots 0a_0)_2\right)/2^w$. Note that c_0 must be an integer. We can derive a bound on $|c_0|$. Every digit in D_w has absolute value at most $2^{w-1}-1$, thus

$$\begin{aligned}|(b_{w-1}\ldots b_1b_0)_2| &\le (2^{w-1}-1)(2^w-1), \text{ and}\\ |(00\ldots 0a_0)_2| &\le 2^{w-1}-1.\end{aligned}$$

Combining these two inequalities gives

$$|(b_{w-1}\ldots b_1b_0)_2 - (00\ldots 0a_0)_2| \le (2^{w-1}-1)2^w$$

and thus $|c_0| \le 2^{w-1}-1$, or equivalently, $|c_0| < 2^{w-1}$.

So, we have $(\alpha_1)_2 = (\beta_1)_2 + c_0$. Let $\beta_1{}'$ denote the string returned by ADD-DIGIT(β_1, c_0). Then $(\beta_1{}')_2 = (\beta_1)_2 + c_0$ and, by Lemma 2.8, $\mathsf{wt}(\beta_1{}') \le \mathsf{wt}(\beta_1)+1$.

Now, we come to the end of the proof. We have

$$\begin{aligned}&\mathsf{wt}(\alpha) > \mathsf{wt}(\beta)\\ \implies &\mathsf{wt}(\alpha_1)+1 > \mathsf{wt}(\beta) \qquad (\text{since } \mathsf{wt}(\alpha) = \mathsf{wt}(\alpha_1)+1)\\ \implies &\mathsf{wt}(\alpha_1)+1 > \mathsf{wt}(\beta_1)+2 \qquad (\text{since } \mathsf{wt}(\beta) \ge \mathsf{wt}(\beta_1)+2)\\ \implies &\mathsf{wt}(\alpha_1) > \mathsf{wt}(\beta_1)+1\\ \implies &\mathsf{wt}(\alpha_1) > \mathsf{wt}(\beta_1{}') \qquad (\text{since } \mathsf{wt}(\beta_1)+1 \ge \mathsf{wt}(\beta_1{}')).\end{aligned}$$

But, $(\alpha_1)_2 = (\beta_1{}')_2$ and $(\alpha_1)_2$ is a w-NAF. Thus $(\alpha_1)_2$ is a shorter counterexample, contrary to our choice of $(\alpha)_2$. This proves the result. □

2.4 Length of Minimal Weight Representations

We have already seen that the length of the w-NAF of an integer is at most one digit longer than its binary representation. In this section, we see that this property is actually a consequence of a more general result. We will show that the length of *any* representation in ${D_w}^*$ with a *minimal number of nonzero digits* is at most one digit longer than its binary representation.

Theorem 2.10. *Let $\alpha = a_{\ell-1} \dots a_1 a_0$ be a string in ${D_w}^*$ such that $a_{\ell-1} \neq 0$ and $(\alpha)_2 = n$. If $\mathsf{wt}(\alpha) \leq \mathsf{wt}(\beta)$ for any $\beta \in {D_w}^*$ with $(\beta)_2 = n$ then $\ell \leq \lfloor \lg |n| \rfloor + 2$.*

Proof. We argue by induction on ℓ, the length of $(\alpha)_2$. Note that since $a_{\ell-1}$ is nonzero the length of $(\alpha)_2$ cannot be zero (i.e., $\ell \geq 1$). Also, $a_{\ell-1} \neq 0$ tells us that $n = (\alpha)_2 \neq 0$ and so $\lg |n|$ is defined.

If $\ell = 1$ then

$$\ell = 1 \leq \lfloor \lg |n| \rfloor + 1 < \lfloor \lg |n| \rfloor + 2,$$

and so the result is true.

Suppose now that $\ell > 1$. Let $\alpha_1 = a_{\ell-1} \dots a_2 a_1$, so that $\alpha = \alpha_1 \| a_0$. Note that

$$\frac{n - a_0}{2} = (\alpha_1)_2.$$

Since $(\alpha)_2$ is a minimal Hamming weight representation of n, $(\alpha_1)_2$ must be a minimal Hamming weight representation of $(n - a_0)/2$. The length of $(\alpha_1)_2$ is $\ell - 1$, so by induction we have

$$\begin{aligned} & \ell - 1 \leq \lfloor \lg |(n - a_0)/2| \rfloor + 2 \\ \Longrightarrow\ & \ell - 1 \leq \lfloor \lg |n - a_0| \rfloor - 1 + 2 \\ \Longrightarrow\ & \ell \leq \lfloor \lg |n - a_0| \rfloor + 2. \end{aligned}$$

If $\lfloor \lg |n - a_0| \rfloor \leq \lfloor \lg |n| \rfloor$ then from the previous step we can conclude

$$\ell \leq \lfloor \lg |n| \rfloor + 2$$

which is the result we want. Thus, we can restrict to the case $\lfloor \lg |n - a_0| \rfloor > \lfloor \lg |n| \rfloor$. In this case, it is clear that $a_0 \neq 0$.

The string α contains at least two nonzero digits, namely $a_{\ell-1}$ and a_0. This tells us that $\mathrm{wt}(\alpha) \geq 2$. Because a_0 is nonzero and $n = (\alpha)_2$, we have also that n is odd. The integer n cannot be equal to any of the digits in D_w. To see this, suppose $n \in D_w$. Then the string $\beta = n$ is in ${D_w}^*$ and $(\beta)_2 = n$. However, $\mathrm{wt}(\beta) = 1$ which is less than $\mathrm{wt}(\alpha) \geq 2$, contrary to our hypothesis. Thus, $|n| > 2^{w-1}$. A consequence of this is that n and $n - a_0$ are either both positive or both negative.

We will suppose n is positive (the case where n is negative is argued in the same manner). Now,

$$\begin{aligned} \lfloor \lg n \rfloor < \lfloor \lg(n - a_0) \rfloor &\implies \lfloor \lg n \rfloor + 1 \leq \lfloor \lg(n - a_0) \rfloor \\ &\implies \lfloor \lg n \rfloor + 1 \leq \lg(n - a_0) \\ &\implies 2^{\lfloor \lg n \rfloor + 1} \leq n - a_0, \end{aligned}$$

so we see that the closed interval $[n, 2^{\lfloor \lg n \rfloor + 1}]$ sits inside the closed interval $[n, n - a_0]$. Let $d = 2^{\lfloor \lg n \rfloor + 1} - n$. Since n is odd, d is odd. The intervals $[n, 2^{\lfloor \lg n \rfloor + 1}]$ and $[n, n - a_0]$ have lengths d and $|a_0|$, respectively. By comparing these lengths, we see that $d \leq |a_0| < 2^{w-1}$. Thus, $d \in D_w$.

Consider the string $\beta \in {D_w}^*$ where

$$\beta = \underbrace{100\ldots0d}_{\lfloor \lg n \rfloor + 2}.$$

Then, $(\beta)_2 = 1 \cdot 2^{\lfloor \lg n \rfloor + 1} + d \cdot 2^0 = n$. So, n can be represented using only 2 nonzero digits. Thus, $\mathrm{wt}(\alpha) \leq 2$. Now, both $\mathrm{wt}(\alpha) \geq 2$ and $\mathrm{wt}(\alpha) \leq 2$, and so $\mathrm{wt}(\alpha) = 2$. Thus, the string α has the following form

$$\alpha = a_{\ell-1}00\ldots0a_0.$$

Now, $(\alpha)_2 = n$ and so $a_{\ell-1} \cdot 2^{\ell-1} + a_0 = n$. Since n is positive and $\lfloor \lg n \rfloor < \lfloor \lg(n - a_0) \rfloor$ it must be that a_0 is negative. However, if a_0 is negative, then $a_{\ell-1}$ must be positive. Thus,

$$\begin{aligned} & a_{\ell-1} \cdot 2^{\ell-1} = n - a_0 \\ \Longrightarrow\ & 2^{\ell-1} \leq n - a_0 \\ \Longrightarrow\ & \ell - 1 \leq \lfloor \lg(n - a_0) \rfloor \\ \Longrightarrow\ & \ell \leq \lfloor \lg(n - a_0) \rfloor + 1. \end{aligned}$$

If $\lfloor \lg(n - a_0) \rfloor \leq \lfloor \lg n \rfloor + 1$, then this gives us $\ell \leq \lfloor \lg n \rfloor + 2$ which is the desired result. To finish the proof, we show that $\lfloor \lg(n - a_0) \rfloor > \lfloor \lg n \rfloor + 1$ would lead to a contradiction. Observe,

$$\begin{aligned} & \lfloor \lg n \rfloor + 1 < \lfloor \lg(n - a_0) \rfloor \\ \Longrightarrow\ & \lg n < \lfloor \lg(n - a_0) \rfloor \quad (\text{since } x < \lfloor x \rfloor + 1) \\ \Longrightarrow\ & \lg n + 1 \leq \lg(n - a_0) \\ \Longrightarrow\ & 2n \leq n - a_0 \\ \Longrightarrow\ & n \leq -a_0 \\ \Longrightarrow\ & n \in D_w \quad (\text{since } n \text{ is odd}), \end{aligned}$$

contradicting the fact that $\mathsf{wt}(\alpha) = 2$. This concludes our proof. □

Note that Proposition 2.5, which we proved directly in Section 2.2, can now be obtained as a consequence of Theorem 2.9 and Theorem 2.10.

2.5 Colexicographic Characterization

For an integer n, consider the set of all representations of n with digits in D_w. We can compare representations in this set in a number of ways. For example, we can order representations according to how many nonzero digits they have. By Theorem 2.9, we know that the w-NAF is a minimal representation under this order, but it is not necessarily unique in this respect. For example, when $w = 3$, the 3-NAF of 5 is $(100\bar{3})_2$ which has two nonzero digits, and so too do the representations $(101)_2, (13)_2$ and $(3\bar{1})_2$.

However, there is another comparison we can make between representations which does, in fact, uniquely identify the w-NAF. This comparison is based on the *position* of nonzero digits and we introduce it now.

Each representation of n identifies a unique finite length string in ${D_w}^*$ with no leading zero digits. If $(\alpha)_2 = n$ and $\alpha \in {D_w}^*$, then we can assume, without loss of generality, that α has no leading zero digits. From the string α, we derive a binary string, $\mathsf{char}(\alpha)$. The string $\mathsf{char}(\alpha)$ is defined as follows: if $\alpha = \ldots a_2 a_1 a_0$ then $\mathsf{char}(\alpha) = \ldots {a_2}' {a_1}' {a_0}'$ where

$$
{a_i}' := \begin{cases} 0 & \text{if } a_i = 0 \\ 1 & \text{otherwise.} \end{cases} \tag{2.6}
$$

For example, if $\alpha = 300\bar{3}0$ then $\mathsf{char}(\alpha) = 10010$. For $\alpha, \beta \in {D_w}^*$ we write $\alpha \preceq \beta$ if $\mathsf{char}(\alpha)$ is less than or equal to $\mathsf{char}(\beta)$ when they are compared *colexicographically*.

The difference between colexicographic order and the more familiar lexicographic order involves the direction in which the characters of a string are read. In lexicographic order, we compare two strings by reading their characters *left-to-right*. In colexicographic order, we compare two strings by reading their characters *right-to-left*. For example, here are five strings sorted lexicographically and colexicographically:

`alfred`	`alfred`
`doug`	`guang`
`edlyn`	`doug`
`guang`	`edlyn`
`jerry`	`jerry`

The relation "$\preceq$" can be used to order the set of representations of n with digits in D_w. For example, suppose $w = 3$ and $n = 42$. Below, we list a number of strings in ${D_3}^*$ which correspond to representations of 42. For each of these strings, we give its associated binary string. This list is sorted under the relation "$\preceq$".

```
     3  0  0 -3  0           1  0  0  1  0
  1  1  0  0 -3  0        1  1  0  0  1  0
  3 -3  0  0 -3  0        1  1  0  0  1  0
  1  0  1  0  1  0        1  0  1  0  1  0
     1  3  0  1  0           1  1  0  1  0
     3 -1  0  1  0           1  1  0  1  0
  1  0  0  3 -1  0        1  0  0  1  1  0
     3  0 -3  3  0           1  0  1  1  0
     3  0 -1 -1  0           1  0  1  1  0
        3  3  3  0              1  1  1  0
```

Notice that the 3-NAF of 42 is the unique smallest representation of the ones considered. Even if we considered *all* the representations of 42 with digits in D_3, the 3-NAF of 42 would still be the unique smallest representation. This result is true in general and is proven in Theorem 2.11.

Theorem 2.11. *Let n be an integer. Of all the representations of n with digits in D_w, the w-NAF of n is the unique smallest representation under the order $\preceq$.*

Proof. When $n = 0$, the only representation of n with digits in D_w is the all-zero representation. The all-zero representation is the w-NAF of 0, so the result is true for $n = 0$. Suppose the result is false for some $n \neq 0$. Choose n so that the length of the w-NAF of n is minimal. Let $(\alpha)_2$ be the w-NAF of n. There is some string $\beta \in {D_w}^*, \beta \neq \alpha$, such that $n = (\beta)_2$ and $\beta \preceq \alpha$.

Recall the definition of α' and β' from (2.6). If n is even then ${a_0}' = {b_0}' = 0$ and so the result is also false for $n/2$, contrary to our choice of n. Thus, n is odd and so ${a_0}' = {b_0}' = 1$. Since α is a w-NAF, ${a_{w-1}}' = {a_{w-2}}' = \cdots = {a_1}' = 0$, and since $\beta \preceq \alpha$, we have ${b_{w-1}}' = {b_{w-2}}' = \cdots = {b_1}' = 0$. Thus

$$\beta = \ldots b_w 00 \ldots 0 b_0$$
$$\alpha = \ldots a_w 00 \ldots 0 a_0.$$

Since $(\alpha)_2 = (\beta)_2$, we have $a_0 \equiv b_0 \pmod{2^w}$. However, $a_0, b_0 \in D_w$, so it must be that $a_0 = b_0$. But this contradicts our choice of n since we see the result is also false for $(n - a_0)/2^w$. Hence, we have the desired result. □

2.6 Comments and Further Work

A detailed discussion of the costs and benefits of using the w-NAF window method for elliptic curve scalar multiplication, including several examples applied to the NIST recommended elliptic curves, is given in [14, Ch. 3]. Much of this analysis is based on the fact that the average density of nonzero digits among all w-NAFs of length ℓ is approximately $1/(w+1)$ (a proof of a similar result is given in [8]). Because of Theorem 2.9, we now know that no other family of D_w-radix 2 representations can have density lower than that of the w-NAF.

As with the result on the length of the w-NAF, the result presented in Lemma 2.4 can be generalized to any minimal weight D_w-radix 2 representation.

The w-NAF window method for scalar multiplication is described in [5] and [41] as a *left-to-right* method[1]. However, Algorithm 2.2 computes the w-NAF of an integer from right to left. This means that the w-NAF of n must first be computed and stored in its entirety before computations to determine nP can begin. There are other representations which use the same digits as the w-NAF but can be deduced from left to right [1, 33]. Like the w-NAF, these new representations have a minimal number of nonzero digits. These representations result in a left-to-right window method which uses less memory.

The results of this chapter further strengthen the analogy between the ordinary NAF and the w-NAF. Another property of the ordinary NAF was recently shown to carry over to the w-NAF. In [23], a simple algorithm is described (due to Chang and Tsao-Wu [7]) which constructs the NAF of n by subtracting the binary representation of n from the binary representation of $3n$. A similar construction has been discovered for the w-NAF [16, 36]. Lemma 2.4 and Proposition 2.5 appear to be very natural consequences of

[1]The w-NAF can also be employed in a right-to-left method for scalar multiplication. More details on this can be found in [28]. However, the left-to-right method is usually preferred over the right-to-left method since it saves a few operations.

this construction.

Theorem 2.9 was proven independently by Avanzi [1]. Avanzi's proof considers radix 2 representations where the nonzero digits have absolute value at most 2^{w-1} and can be either odd or even. Even when these additional digits are included, Avanzi's argument shows that the w-NAF still has minimal weight.

Chapter 3

Left to Right Construction

3.1 Window Methods

A family of algorithms, known as *window methods*, use the approach described in Algorithm 3.1 to perform scalar multiplication.

Algorithm 3.1: RADIX-2-WINDOW-METHOD(n, P)

```
fix a set of digits, $D \subset \mathbb{Z}$.
for each $d \in D$ with $d > 0$
  do $P_d \leftarrow dP$
compute & store a representation $(a_{\ell-1} \dots a_1 a_0)_2 = n$ with $a_i \in D$.
$Q \leftarrow \infty$
for $i = \ell - 1 \dots 0$
  do $Q \leftarrow 2Q$
     if $a_i \neq 0$
       then if $a_i > 0$
              then $Q \leftarrow Q + P_{a_i}$
              else $Q \leftarrow Q - P_{-a_i}$
return $Q$
```

For example, suppose $D = \{0, \pm 1, \pm 3\}$. Then Algorithm 3.1 first computes and stores P and $3P$. After a D-radix 2 representation of n is computed its digits are read from left to right by the "for" loop and nP is computed using doubling, addition and subtraction operations. Including negative digits in D takes advantage of the fact that subtracting an elliptic curve point can be done just as efficiently as adding it. A D-radix 2 representation of n can be computed by sliding a window of width 3 from right to left across the $\{0,1\}$-radix 2 representation of n (see Section 3.3).

Blake, Seroussi and Smart [5], Cohen, Miyaji and Ono [9] and Solinas [41] independently suggested a specialization of Algorithm 3.1 called the *width-w nonadjacent form window method* (this terminology is due to Solinas).

Let D_w be the set of w-NAF digits; that is,

$$D_w := \{0\} \cup \{d \in \mathbb{Z} : d \text{ odd}, |d| < 2^{w-1}\}.$$

If, in Algorithm 3.1, the digit set D_w is used and the representation $(a_{\ell-1} \dots a_1 a_0)_2$ is always chosen to be a w-NAF, then this is the w-NAF window method.

One advantage of using the w-NAF of an integer is that it has a *minimal number of nonzero digits* (see Chapter 2). A nonzero integer has an infinite number of D_w-radix 2 representations and any of these representations could be used in Algorithm 3.1. However, the choice of representation affects the performance of the algorithm. In the "for" loop, an addition/subtraction operation is performed for every nonzero digit of $(a_{\ell-1} \dots a_1 a_0)_2$. It is thus desirable to use a D_w-radix 2 representation of n with as few nonzero digits as possible. No other D_w-radix 2 representation of an integer has fewer nonzero digits than its w-NAF. [1]

The w-NAF of an integer is computed by sliding a window of width w from right to left across the $\{0,1\}$-radix 2 representation of n. This procedure deduces the digits of the w-NAF from right to left; however, the "for" loop of Algorithm 3.1 reads these digits from left to right. This means that the w-NAF of n must be computed and stored in its entirety before computations inside the "for" loop can begin.

[1]Cohen [8] presents a detailed average case analysis of the w-NAF window method.

This problem of opposing directions occurs in many window methods and has been lamented by both Müller [34] and Solinas [41]. If the algorithm which computes the D_w-radix 2 representation of n worked in the same direction as the "for" loop, Algorithm 3.1 could be modified so that it uses less memory. In that case, it would be unnecessary to store the representation $(a_{\ell-1} \ldots a_1 a_0)_2$ since its digits could be computed inside the "for" loop as they are needed. This savings is most relevant for memory constrained devices like smartcards.

We propose a new family of D_w-radix 2 representations and prove that, like the w-NAFs, these representations have a minimal number of nonzero digits. The digits of these representations can be deduced from left to right and thus can be used to reduce the memory requirements of Algorithm 3.1.

Joye and Yen [18] give a very simple left-to-right algorithm for computing the digits of a $\{0, \pm 1\}$-radix 2 representation of an integer. They also prove that the representations constructed by this algorithm have a minimal number of nonzero digits. Their results apply to the digit set D_2, whereas ours apply to arbitrary D_w with $w \geq 2$.

The outline of the chapter is as follows. In Section 3.2, we introduce the algorithm used to construct our representations and then, in Section 3.3, describe how it can be efficiently implemented. In Section 3.4, we prove minimality. We end with a discussion of related work and some remarks.

3.2 The Algorithm

We introduce an algorithm for computing the digits of a D_w-radix 2 representation from left to right. Our discussion is mainly intended to illustrate the idea behind the algorithm. Details about how the algorithm can be efficiently implemented are postponed until Section 3.3.

In order to motivate our algorithm, we begin by describing a simple method of computing the $\{0, 1\}$-radix 2 representation of a positive integer. This could be used if an integer were represented in some other way; for example, if we wanted to convert from radix 10 to radix 2.

Suppose we want to deduce the digits of the $\{0,1\}$-radix 2 representation of 233 from left to right. This is easily done by subtracting powers of 2. The number $n = 233$ is small enough so that we can quickly determine $2^{\lfloor \lg n \rfloor}$; this is the power of 2 closest to, but not larger than, n. Once we determine $2^{\lfloor \lg n \rfloor}$, we replace n with $n - 2^{\lfloor \lg n \rfloor}$ and then repeat these steps until we reach 0. Doing so gives us

n	$2^{\lfloor \lg n \rfloor}$
233	$128 = 2^7$
105	$64 = 2^6$
41	$32 = 2^5$
9	$8 = 2^3$
1	$1 = 2^0$.

Thus, we see that $233 = 2^7 + 2^6 + 2^5 + 2^3 + 2^0 = (11101001)_2$.

We can modify this process so that it returns a $\{0,1\}$-string. We begin with a string, α, which is initially empty. In each step, we append to α a (possibly empty) run of 0's followed by a single 1. Doing so gives us

n	$2^{\lfloor \lg n \rfloor}$	α
233	$128 = 2^7$	$\alpha \parallel 1$
105	$64 = 2^6$	$\alpha \parallel 1$
41	$32 = 2^5$	$\alpha \parallel 1$
9	$8 = 2^3$	$\alpha \parallel 01$
1	$1 = 2^0$	$\alpha \parallel 001$.

Note that the symbol $\parallel$ denotes concatenation. When n reaches 0, α is equal to 11101001 and we see that $(\alpha)_2 = 233$.

For an arbitrary nonnegative integer, we can describe this process in pseudocode. Let $D = \{0,1\}$ and define

$$\mathcal{C} := \{d \cdot 2^i : d \in D \setminus \{0\}, i \in \mathbb{Z}, i \geq 0\}.$$

The set $\mathcal{C}$ consists of all the positive powers of 2. Here is a description of the procedure:

```
$\alpha \leftarrow \epsilon$
**while** $n \neq 0$
    **do** { $c \leftarrow$ an element in $\mathcal{C}$ closest to, but not larger than, $n$
           append digits to $\alpha$ according to the value of $c$
           $n \leftarrow n - c$
**return** $\alpha$
```

Note that ϵ denotes the empty string. The set $\mathcal{C}$ is infinite, however this is not a concern since we do not need to store $\mathcal{C}$. To choose an element in $\mathcal{C}$ closest to, but not larger than, n, we simply compute $2^{\lfloor \lg n \rfloor}$.

Consider now the digit set $D_2 = \{0, \pm 1\}$. We would like to somehow deduce the digits of a D_2-radix 2 representation of an integer from left to right. We can do this by modifying our previous procedure slightly. We first define

$$\mathcal{C}_2 := \{d \cdot 2^i : d \in D_2 \setminus \{0\}, i \in \mathbb{Z}, i \geq 0\}.$$

Note that $\mathcal{C}_2$ consists of the positive and negative powers of 2. Now consider the following procedure:

```
$\alpha \leftarrow \epsilon$
**while** $n \neq 0$
    **do** { $c \leftarrow$ an element in $\mathcal{C}_2$ closest to $n$
           append digits to $\alpha$ according to the value of $c$
           $n \leftarrow n - c$
**return** $\alpha$
```

The only change above is that the condition "closest to, but not larger than, n" is now simply "closest to n". If we apply this procedure to $n = 233$ we get

n	c	α
233	$256 = 2^8$	$\alpha \parallel 1$
-23	$-16 = -2^4$	$\alpha \parallel 000\bar{1}$
-7	$-8 = -2^3$	$\alpha \parallel \bar{1}$
1	$1 = 2^0$	$\alpha \parallel 001$.

When n reaches 0, α is equal to $1000\overline{1}\overline{1}001$ and we see that

$$(\alpha)_2 = (1000\overline{1}\overline{1}001)_2 = 2^8 - 2^4 - 2^3 + 2^0 = 233.$$

This same example is worked by Joye and Yen [18] and our representation is identical to theirs. Note also that, as in the previous case, the set $\mathcal{C}_2$ does not need to be stored. To choose a closest element from $\mathcal{C}_2$, we compute $2^{\lfloor \lg |n| \rfloor}$ and then compare $|n|$ to $2^{\lfloor \lg |n| \rfloor}$ and $2^{\lfloor \lg |n| \rfloor + 1}$.

In the general case, we would like to construct D_w-radix 2 representations from left to right for arbitrary $w \geq 2$. Here is a procedure which does so:

```
Algorithm 3.2: MSF_w(n)

comment: $w \geq 2$, $D_w = \{0\} \cup \{d \in \mathbb{Z} : d \text{ odd}, |d| < 2^{w-1}\}$, and
         $\mathcal{C}_w = \{d \cdot 2^i : d \in D_w \setminus \{0\}, i \in \mathbb{Z}, i \geq 0\}$

$\alpha \leftarrow \epsilon$
while $n \neq 0$
  do { $c \leftarrow$ an element in $\mathcal{C}_w$ closest to $n$
       append digits to $\alpha$ according to the value of $c$
       $n \leftarrow n - c$
return $\alpha$
```

As before, the set $\mathcal{C}_w$ does not need to be stored. We will see in Section 3.3 that this procedure can be implemented efficiently by sliding a window of width $w+1$ from left to right across the $\{0,1\}$-radix 2 representation of n. A description of how digits are appended to α will be provided shortly. We call this procedure $\text{MSF}_w(n)$.

We have given this procedure the title "Algorithm". To justify this we must show that $\text{MSF}_w(n)$ terminates for all $n \in \mathbb{Z}$. If $n = 0$, then $\text{MSF}_w(n)$ clearly terminates, so we need only consider $n \neq 0$. To finish the argument we need a Lemma.

Lemma 3.3. *Let n be a nonzero integer. If c is an element in $\mathcal{C}_w$ closest to n, then*

$$|n - c| \leq \frac{2^{\lfloor \lg |n| \rfloor}}{2^{w-1}}.$$

Proof. We will assume $n > 0$; the proof for $n < 0$ is similar. First, suppose $n < 2^{w-1}$. If n is odd then $n \in D_w$ and hence $n = n \cdot 2^0 \in \mathcal{C}_w$. If n is even then $n = d \cdot 2^i$ for some $i < w - 1$ and $d \in D_w$ and hence $n = d \cdot 2^i \in \mathcal{C}_w$. Thus, if $n < 2^{w-1}$ and c is closest to n then $c = n$. So, $n - c = 0$ and the inequality is valid.

Consider the closed interval $[2^{w-2}, 2^{w-1}]$. This interval contains exactly $2^{w-2} + 1$ integers and, as we have just seen, each one is in $\mathcal{C}_w$. More generally, for $i \geq w-2$, the interval $[2^i, 2^{i+1}]$ contains exactly $2^{w-2}+1$ elements of $\mathcal{C}_w$. Moreover, these elements partition the interval into 2^{w-2} subintervals of equal length.

Now, suppose $n \geq 2^{w-1}$. The integer n lies in the interval $[2^{\lfloor \lg n \rfloor}, 2^{\lfloor \lg n \rfloor + 1}]$ which has length $2^{\lfloor \lg n \rfloor}$. Since $\lfloor \lg n \rfloor \geq w - 1$, this interval contains exactly $2^{w-2} + 1$ elements of $\mathcal{C}_w$. These elements partition this interval into subintervals of length $2^{\lfloor \lg n \rfloor}/2^{w-2}$. The integer n lies in a subinterval and one endpoint of this subinterval is c. Since both endpoints are in $\mathcal{C}_w$, we have

$$|n - c| \leq 1/2 \cdot (2^{\lfloor \lg n \rfloor}/2^{w-2}) = 2^{\lfloor \lg n \rfloor}/2^{w-1}.$$

Since n is positive, we have that $|n| = n$, and this gives the desired result. □

To show that $\mathrm{MSF}_w(n)$ terminates for $n \neq 0$, it suffices to show that $|n| > |n - c|$. Suppose to the contrary that $|n| \leq |n - c|$. Then

$$\begin{aligned} & |n| \leq |n - c| \\ \implies & |n| \leq 2^{\lfloor \lg |n| \rfloor}/2^{w-1} \qquad \text{(by Lemma 3.3)} \\ \implies & 2^{\lfloor \lg |n| \rfloor} \leq 2^{\lfloor \lg |n| \rfloor}/2^{w-1} \\ \implies & 1 \leq 1/2^{w-1} \\ \implies & w \leq 1. \end{aligned}$$

However, this is a contradiction because $w \geq 2$. So, the sequence formed by taking the absolute value of the variable n during the execution of $\mathrm{MSF}_w(n)$ is strictly decreasing. Thus, the variable n must reach 0, and so $\mathrm{MSF}_w(n)$ terminates for all $n \in \mathbb{Z}$.

The string α returned by $\mathrm{MSF}_w(n)$ has been defined somewhat informally. We present a more rigorous definition based on the values that the variable c assumes during the execution of $\mathrm{MSF}_w(n)$. For an input, n, we define $\alpha = \ldots a_2 a_1 a_0$ to be the string such that

$$a_i := \begin{cases} d & \text{if } c \text{ assumes the value } d \cdot 2^i \text{ at some point in the algorithm,} \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

Clearly, each $a_i \in D_w$, and so α is a D_w-string. There is, however, one possible problem with this definition. Suppose c assumes the two distinct values $d_0 \cdot 2^i$ and $d_1 \cdot 2^i$ which share the same power of 2. In that case, the value of a_i is undefined. Fortunately, this problem never occurs, as is shown in the following Lemma.

Lemma 3.4. *Let c_0, c_1 and n be nonzero integers such that c_0 is an element in $\mathcal{C}_w$ closest to n and c_1 is an element in $\mathcal{C}_w$ closest to $n - c_0$. If $c_0 = d_0 2^{i_0}$ and $c_1 = d_1 2^{i_1}$ with $d_0, d_1 \in D_w$, then $i_0 > i_1$.*

Proof. We begin by bounding $|c_0|$. Since $|n|$ lies in the interval $[2^{\lfloor \lg |n| \rfloor}, 2^{\lfloor \lg |n| \rfloor + 1}]$ and both endpoints of this interval are in $\mathcal{C}_w$ we have

$$\begin{aligned} 2^{\lfloor \lg |n| \rfloor} &\leq |c_0| \leq 2^{\lfloor \lg |n| \rfloor + 1} \\ \implies 2^{\lfloor \lg |n| \rfloor} &\leq |d_0| \, 2^{i_0} \leq 2^{\lfloor \lg |n| \rfloor + 1}. \end{aligned}$$

Thus,

$$\begin{aligned} |d_0| = 1 &\implies i_0 = \lfloor \lg |n| \rfloor \text{ or } \lfloor \lg |n| \rfloor + 1, \text{ and} \\ |d_0| > 1 &\implies i_0 = \lfloor \lg |n| \rfloor - \lfloor \lg |d_0| \rfloor . \end{aligned}$$

Since $d_0 \in D_w$, $|d_0| < 2^{w-1}$ and thus $\lfloor \lg |d_0| \rfloor \in \{0, 1, \ldots w-2\}$. From these implications we can conclude that

$$i_0 \in \{ \lfloor \lg |n| \rfloor - w + 2, \ldots, \lfloor \lg |n| \rfloor + 1 \}.$$

In the same way, we can deduce that

$$i_1 \in \{ \lfloor \lg |n - c_0| \rfloor - w + 2, \ldots, \lfloor \lg |n - c_0| \rfloor + 1 \}.$$

Now, by Lemma 3.3, we have

$$\begin{aligned} & |n - c_0| \leq 2^{\lfloor \lg |n| \rfloor - w + 1} \\ \Longrightarrow\ & 2^{\lfloor \lg |n - c_0| \rfloor} \leq 2^{\lfloor \lg |n| \rfloor - w + 1} \\ \Longrightarrow\ & \lfloor \lg |n - c_0| \rfloor \leq \lfloor \lg |n| \rfloor - w + 1 \\ \Longrightarrow\ & \lfloor \lg |n - c_0| \rfloor + 1 \leq \lfloor \lg |n| \rfloor - w + 2. \end{aligned}$$

Thus, the largest possible value of i_1 is less than or equal to the smallest possible value of i_0. Hence $i_1 \leq i_0$.

To see that $i_1 \neq i_0$, first note that

$$|n| > |n - c_0| > |(n - c_0) - c_1|,$$

and so

$$|n - c_0| > |n - (c_0 + c_1)|,$$

thus the integer $c_0 + c_1$ is closer to n than c_0 is. If $i_1 = i_0$, then

$$c_0 + c_1 = (d_0 + d_1)2^{i_0}.$$

Since d_0 and d_1 are both odd, $d_0 + d_1 = d \cdot 2^i$ for some $d \in D_w \setminus \{0\}$ and $i \geq 1$. Thus, $c_0 + c_1 \in \mathcal{C}_w$. However, this contradicts the fact that c_0 is closest to n. So it must be that $i_1 \neq i_0$, and hence $i_1 < i_0$, as required. $\square$

By Lemma 3.4, the string α is well defined. Moreover, as we saw in our earlier examples, Lemma 3.4 tells us that α can be constructed using operations of the form

$$\alpha \leftarrow \alpha \parallel 0^t d \quad \text{where} \quad t \geq 0,\ d \in D_w,\ d > 0.$$

Actually, we need to be a bit more precise here. If n is odd, then α can be constructed using only operations like the one above; however, if n is even, then α will need to have a run of zeros appended to it before it is returned.

From the definition given in (3.1) we can now show that the string returned by $\mathrm{MSF}_w(n)$ is in fact a D_w-radix 2 representation of n (i.e., the algorithm is correct). Let S be the set of values that the variable c assumes during the execution of $\mathrm{MSF}_w(n)$. For $\alpha = \ldots a_1 a_0$ we have

$$(\alpha)_2 = \sum_{i \geq 0} a_i 2^i = \sum_{\substack{i \\ a_i \neq 0}} a_i 2^i = \sum_{c \in S} c = n.$$

The representation returned by $\mathrm{MSF}_w(n)$ for a given value of n is not necessarily unique. For example, when $w = 3$, $D_3 = \{0, \pm 1, \pm 3\}$ and for $n = 5$ we see that both $4 = 2^2$ and $6 = 3 \cdot 2^1$ are elements in $\mathcal{C}_3$ closest to 5. Thus, $\mathrm{MSF}_3(5)$ will return one of the representations

$$(101)_2 \quad \text{or} \quad (3\bar{1})_2.$$

From the description of Algorithm 3.2, it is apparent that $\mathrm{MSF}_w(n)$ will have more than one possible output only when some value of the variable n has more than one closest element in $\mathcal{C}_w$. This occurs only when a value of the variable n is the midpoint between neighbouring elements of $\mathcal{C}_w$.

We argue that there are at most two distinct outputs of $\mathrm{MSF}_w(n)$ for any $n \in \mathbb{Z}$. Imagine a list of outputs of $\mathrm{MSF}_w(n)$. Let i_0 be the largest value of i such that two outputs differ at digit i. If i_0 does not exist then all the outputs are the same; otherwise, let $\alpha = \ldots a_2 a_1 a_0$ and $\beta = \ldots b_2 b_1 b_0$ be two outputs with $a_{i_0} \neq b_{i_0}$. We have

$$n = (\ldots a_{i_0} \ldots a_1 a_0)_2 = (\ldots b_{i_0} \ldots b_1 b_0)_2.$$

Let

$$n' = (a_{i_0} \ldots a_1 a_0)_2 = (b_{i_0} \ldots b_1 b_0)_2.$$

At least one of a_{i_0} and b_{i_0} is nonzero. We assume, without loss of generality, that $a_{i_0} \neq 0$. Let $c_0 = a_{i_0} 2^{i_0}$. The value c_0 is an element in $\mathcal{C}_w$ is closest to n'. Since $a_{i_0} \neq b_{i_0}$ there must be another value, say c_1, closest to n', and this value must be encoded as the most significant nonzero digit of $(b_{i_0} \ldots b_1 b_0)_2$. Since both c_0 and c_1 are closest to n', n' must be the midpoint between c_0 and c_1. Thus, $|n' - c_0|$ is as large as possible, so by Lemma 3.3 we have

$$\left| n' - c_0 \right| = \left| n' - c_1 \right| = 2^{\lfloor \lg |n'| \rfloor - w + 1}.$$

Let $t = \lfloor \lg |n'| \rfloor - w + 1$. Since $n' - c_0 = \pm 2^t$, $n' - c_0 \in \mathcal{C}_w$ and $n' - c_0$ is the unique element in $\mathcal{C}_w$ closest to $n' - c_0$. Similarly, $n' - c_1$ is the unique element in $\mathcal{C}_w$ closest to $n' - c_1$. Thus, the least significant nonzero digits of α and β correspond to the values $n' - c_0$ and $n' - c_1$; that is, the least significant nonzero digits of α and β are a_t and b_t where one of a_t, b_t is 1 and the other -1, so

$$n' = c_0 + 2^t = c_1 - 2^t \quad \text{or} \quad n' = c_0 - 2^t = c_1 + 2^t.$$

Note that the example above demonstrates this property. Thus, there are just two kinds of outputs: ones that encode c_0 and ones that encode c_1.

From the preceding discussion, we can derive the following Lemma:

Lemma 3.5. *Let α and β be two outputs of $MSF_w(n)$. Then α and β have the same number of nonzero digits.*

Proof. If $\alpha = \beta$ then there is nothing to prove, so we can assume $\alpha \neq \beta$. Let

$$\alpha = \ldots a_{i_0} \ldots a_1 a_0 \quad \text{and} \quad \beta = \ldots b_{i_0} \ldots b_1 b_0,$$

where i_0 is the largest value of i such that $a_i \neq b_i$. From our discussion above, the strings $a_{i_0} \ldots a_1 a_0$ and $b_{i_0} \ldots b_1 b_0$ each contain exactly two nonzero digits. Thus α and β have the same number of nonzero digits. $\square$

It is possible to implement Algorithm 3.2 in such a way that it returns a unique representation for every $n \in \mathbb{Z}$. For example, if c_0 and c_1 are both closest to n then we might resolve this ambiguity by choosing the larger one. Because of Lemma 3.5, we know that, however Algorithm 3.2 is implemented, all outputs will have the same number of nonzero digits (for a given input). In fact, any representation of n constructed by Algorithm 3.2 will have a *minimal number of nonzero digits*, and we will prove this in Section 3.4. In the next section, we describe how to implement Algorithm 3.2 efficiently.

3.3 Implementations

We first review a known right-to-left sliding window method for constructing D_w-radix 2 representations. Then we describe how Algorithm 3.2 can be implemented using a left-to-right sliding window method. We also give a new implementation of Algorithm 3.1 which incorporates our left-to-right representations.

3.3.1 Right-to-Left

Suppose that we want to deduce a radix 2 representation of the integer 379 using the digits $D_3 = \{0, \pm 1, \pm 3\}$. If we know the $\{0, 1\}$-radix 2 representation of 379 then this is easily done. Consider the following table

β	c	β'	c'
001	0	001	0
011	0	003	0
101	0	$00\bar{3}$	1
111	0	$00\bar{1}$	1
000	1	001	0
010	1	003	0
100	1	$00\bar{3}$	1
110	1	$00\bar{1}$	1

This table describes a map, $(\beta, c) \mapsto (\beta', c')$, between ordered pairs. The ordered pairs consist of a 3-digit string and a *carry*, c. Notice that for each row of the table, the string β' corresponds to $((\beta)_2 + c)$ mods 2^3. After initializing the carry to 0, we can apply these transformations by sliding a 3-digit *window* from right to left across the $\{0, 1\}$-radix 2 representation:

$$\begin{array}{r|r}
379 = (0101111\underline{01\overset{0}{1}})_2 & 003 \\
(01011\underline{1\overset{0}{1}1}011)_2 & 00\bar{1}003 \\
(0\underline{1\overset{1}{0}1}111011)_2 & 000\bar{1}003 \\
(\underline{0\overset{1}{1}0}1111011)_2 & 003000\bar{1}003
\end{array}$$

Each time the contents of the window and the value of the carry match an entry in the left hand column of the table we output the corresponding 3-digit string, update the carry and then advance the window 3 digits to the left. Otherwise, we output a single 0, leave the carry unchanged and advance the window 1 digit to the left.

This process constructs an integer's 3-NAF, and it does so using only a look-up table. If we allow simple bit operations, like xor, the number of rows in the table can be halved.

3.3.2 Left-to-Right

When $w = 3$ we have $\mathcal{C}_3 = \{d \cdot 2^i : d \in D_3 \setminus \{0\}, i \in \mathbb{Z}, i \geq 0\}$. The first few positive elements of $\mathcal{C}_3$ are

$$1, \underline{2}, 3, \underline{4}, 6, \underline{8}, 12, \underline{16}, 24, \underline{32}, 48, \underline{64}, 96, \underline{128}, 192, \underline{256}, 384, \underline{512} \ldots$$

Notice that for $i \geq 2$, the intervals $[2^{i-1}, 2^i]$ (the end points of which are underlined above) each contain exactly 3 elements of $\mathcal{C}_3$. Consider the integer 379. From the list of values above, we see that 384 is the element in $\mathcal{C}_3$ closest to 379, however this can also be determined from the $\{0,1\}$-radix 2 representation of 379.

We first determine two neighbouring elements of $\mathcal{C}_3$, call them c' and c'', such that $379 \in [c', c'']$. The value c' is the unique element in $\mathcal{C}_3$ closest to, but not larger than, 379. If $379 = (0b_{\ell-1}b_{\ell-2}\ldots b_1b_0)_2$ with $b_i \in \{0,1\}$ and $b_{\ell-1} = 1$, we can determine the value of c' by simply reading the 3 digit prefix of this representation (i.e., $0b_{\ell-1}b_{\ell-2}$). If the prefix is 010, then $c' = (010)_2 \cdot 2^{\ell-2} = 2^{\ell-1}$. If the prefix is 011, then $c' = (011)_2 \cdot 2^{\ell-2} = 3 \cdot 2^{\ell-2}$. Since $379 = (0101111011)_2$, we see that $c' = (010)_2 \cdot 2^7 = 256$.

The most significant nonzero digit of the representation $379 = (0101111011)_2$ tells us that $2^8 \leq 379 < 2^9$. Since 2^8 and 2^9 are both in $\mathcal{C}_3$, it must be that$[c', c''] \subseteq [2^8, 2^9]$. The interval $[2^8, 2^9]$ has length 2^8 and contains exactly three elements of $\mathcal{C}_3$, thus the interval $[c', c'']$ must have length $2^8/2 = 2^7$. So, we see that $c'' = c' + 2^7 = 256 + 128 = 384$.

We have deduced that $379 \in [256, 384]$ where 256 and 384 are neighbouring elements of $\mathcal{C}_3$. Now, the question is, which of $256, 384$ is closer to 379. This is determined by the digit immediately to the right of the 3 digit prefix we considered above. If this digit is 0, 256 is closest, otherwise 384 is closest. Since $379 = (010\underline{1}111011)_2$ we see that 384 is the element in $\mathcal{C}_3$ closest to 379.

To continue building a representation of 379 using Algorithm 3.2, we must now determine an element closest to $379 - 384 = -5$. Again, we can use the $\{0,1\}$-radix 2 representation of 379 to make this determination.

Suppose we have $-5 = (\bar{1}b_{\ell-1}b_{\ell-2}\ldots b_1b_0)_2$ with $b_i \in \{0,1\}$ and $b_{\ell-1} = 0$. Then, as before, we can determine neighbouring elements of $\mathcal{C}_3$, c' and c'', such that $-5 \in [c', c'']$. The value c' is determined by simply reading the 3 digit prefix of this representation (i.e., $\bar{1}b_{\ell-1}b_{\ell-2}$). If the prefix is $\bar{1}00$, then $c' = (\bar{1}00)_2 \cdot 2^{\ell-2} = -2^{\ell}$. If the prefix is $\bar{1}01$, then $c' = (\bar{1}01)_2 \cdot 2^{\ell-2} = -3 \cdot 2^{\ell-2}$.

It is not difficult to construct such a representation of -5. Observe

$$\begin{aligned} 379 &= (0101111011)_2 \\ 384 &= (0110000000)_2 \\ \implies 379 - 384 &= \quad (\bar{1}1111011)_2. \end{aligned}$$

Since the digits $\bar{1}1$ can be replaced by $0\bar{1}$, we have that $-5 = (\bar{1}011)_2$. Now we see that $c' = (\bar{1}01)_2 \cdot 2 = -3 \cdot 2 = -6$.

The most significant nonzero digit of the representation $-5 = (\bar{1}011)_2$ stands for -2^3. Because the following digit is 0, we have that $-2^3 \leq -5 < -2^2$. Since -2^3 and -2^2 are both in $\mathcal{C}_3$, it must be that$[c', c''] \subseteq [-2^3, -2^2]$. The interval $[-2^3, -2^2]$ has length 2^2 and contains exactly three elements of $\mathcal{C}_3$, thus the interval $[c', c'']$ must have length $2^2/2 = 2$. So, we see that $c'' = c' + 2 = -6 + 2 = -4$.

Now we must decide which of c', c'' is closest to -5. As before, we can determine this by reading the the digit immediately to the right of the 3 digit prefix. If this digit is 0, c' is closest. If this digit is 1, c'' is closest. In this case, both $c' = -6$ and $c'' = -4$ are closest to -5, however, this rule simply distinguishes one of them. Since $-5 = (\bar{1}01\underline{1})_2$ we see that -4 is an element in $\mathcal{C}_3$ closest to -5.

To finish building our representation of 379 we must now determine an element closest to $-5 - (-4) = -1$. Clearly, -1 is the element in $\mathcal{C}_3$ closest to -1, however, we can also make this determination be applying our previous arguments to the representation $-1 = (\bar{1}.000)_2$. We can always examine a 3 digit prefix of a representation by taking zeros from the right of the radix point when necessary.

The techniques we have described for determining closest elements in $\mathcal{C}_3$ can be implemented as a 4-digit window slides left to right across a $\{0, 1\}$-radix 2 representation. Consider the following table

β	β'
0100	010
0101	003
0110	003
0111	100
1000	$\bar{1}00$
1001	$00\bar{3}$
1010	$00\bar{3}$
1011	$0\bar{1}0$

This table describes a map, $\beta \mapsto \beta'$, between strings. The relation between these strings is based on choosing closest elements in $\mathcal{C}_3$.

The first four rows of the table are filled in by determining closest elements to integers represented as $(01b_{\ell-2}b_{\ell-3}\ldots)_2$. The last four rows of the table are filled in by determining closest elements to integers represented as $(\bar{1}0b_{\ell-2}b_{\ell-3}\ldots)_2$. It can be shown that if $n = (\overline{b_\ell}b_{\ell-1}b_{\ell-2}b_{\ell-3}\ldots b_0)_2$ with $b_i \in \{0, 1\}$ and $b_\ell \neq b_{\ell-1}$, then, for the element c closest to n that we choose, we have $n - c = (\overline{b_{\ell-3}}b_{\ell-4}\ldots b_0)_2$.

Returning to our example, $n = 379$, we have

$$\begin{array}{r|l}
379 = (\underline{0101}111011.000)_2 & 003 \\
(010\underline{1111}011.000)_2 & 0030 \\
(0101\underline{1110}11.000)_2 & 00300 \\
(01011\underline{1101}1.000)_2 & 003000 \\
(010111\underline{1011}.000)_2 & 00300001\bar{1}0 \\
(010111101\underline{1.000})_2 & 0030000\bar{1}0\bar{1}.000
\end{array}$$

As with the construction of the 3-NAF of 379, each time the contents of the window match an entry in the left hand column of the table we output the corresponding 3-digit string and then advance the window 3 digits to the right. Otherwise, we output a single 0 and advance the window 1 digit to the right.

If we work from the description of Algorithm 3.2 in Section 3.2, we might construct a different representation of 379 than the one above. Since $379 - 3 \cdot 2^7 = -5$ and -5 has two closest elements in $\mathcal{C}_3$, Algorithm 3.2 might also return $379 = (300000\bar{3}1)_2$ (note that this example demonstrates that, unlike the 3-NAFs, the representations constructed by Algorithm 3.2 can have adjacent nonzero digits). The implementation we have described is deterministic, thus it must somehow distinguish one of two closest elements in $\mathcal{C}_w$. It does so by always selecting a *largest* closest element.

For *general* $w \geq 2$, Algorithm 3.2 can be implemented by sliding a window of width $w+1$ from left to right across the $\{0,1\}$-radix 2 representation of n. This implementation is based on the following facts. If

$$n = (\overline{b_\ell} b_{\ell-1} \ldots b_1 b_0)_2 \text{ with } b_i \in \{0,1\} \text{ and } b_\ell \neq b_{\ell-1}, \tag{3.2}$$

then we can determine $c \in \mathcal{C}_w$ closest to n from the $w+1$ digit string $b_\ell b_{\ell-1} \ldots b_{\ell-w}$. For this value c closest to n, we have

$$n - c = (\overline{b_{\ell-w}} b_{\ell-w-1} \ldots b_1 b_0)_2. \tag{3.3}$$

The resulting look-up table will contain 2^w rows and describes a map from $w+1$ digit strings to w digit strings. Due to the symmetry in the table, if we allow simple bit operations, like xor, the second half of the table does not need to be stored.

3.3.3 A New Window Method

In our example implementation for $w = 3$, our window slides either 3 digits to the right (after the window matches an entry in the table) or one digit to the right (otherwise). This is because the strings output in these cases have length 3 (a string β') or length 1 (a single 0). However, it is not necessary for the strings β' to all have the same length.

If we take our previous table and delete the trailing zeros from each string β' then we get

β	β'
0100	01
0101	003
0110	003
0111	1
1000	$\bar{1}$
1001	$00\bar{3}$
1010	$00\bar{3}$
1011	$0\bar{1}$

β	j	d
0100	2	1
0101	3	3
0110	3	3
0111	1	1
1000	1	$\bar{1}$
1001	3	$\bar{3}$
1010	3	$\bar{3}$
1011	2	$\bar{1}$

In the left table, the strings β' are all of the form $0^{j-1}d$ where d is a nonzero digit in D_3. The right table is just an encoding of the left table. The left table can be used to construct D_3-radix representations similar to the way we described in the previous section. The only difference is the window slides right 1, 2 or 3 digits at a time; the number being equal to the length of the output string (either β' or a single 0).

This implementation of Algorithm 3.2 can be incorporated easily with Algorithm 3.1. From the right table, we can define a function T_3 which maps strings in $\{0,1\}^4$ to ordered pairs, (j,d), with the additional condition that if $\beta \in \{0,1\}^4$ does not appear in the table then $T_3(\beta) = (1,0)$.

Here is the resulting algorithm for scalar multiplication, which works for an arbitrary value of $w \geq 2$:

```
Algorithm 3.6: $w$-MSF-WINDOW-METHOD$(n, P)$

comment: $w \geq 2,\ D_w = \{0\} \cup \{d \in \mathbb{Z} : d \text{ odd}, |d| < 2^{w-1}\}$
         $n = (0b_{\ell-1} \dots b_1 b_0)_2$, where $b_i \in \{0, 1\}$
         $\beta_i = b_i b_{i-1} \dots b_{i-w}$

external $T_w : \beta_i \mapsto (j, d)$

for each $d \in D_w$ with $d > 0$
  do $P_d \leftarrow dP$
$Q \leftarrow \infty,\ i \leftarrow \ell$
while $i \geq 0$
  do
    $(j, d) \leftarrow T_w(\beta_i)$
    $Q \leftarrow 2^j Q$
    if $d \neq 0$
      then
        if $d > 0$
          then $Q \leftarrow Q + P_d$
          else $Q \leftarrow Q - P_{-d}$
    $i \leftarrow i - j$
return $Q$
```

Constructing the function $T_w : \{0,1\}^{w+1} \to \{1, 2, \dots, w\} \times D_w$ for arbitrary $w \geq 2$ is straightforward, provided we already have the width-w look-up table described in Section 3.3.2; we simply delete the trailing zeros on the output strings and then encode the output strings as ordered pairs.

Building the width-w look-up table all at once is not difficult, however, it is also possible to build the table on the fly. An algebraic expression for the element $c \in \mathcal{C}_w$ closest to n can be obtained by subtracting the representation for $n - c$ in (3.3) from that of n in (3.2). The resulting expression for c is

$$c = \left((\overline{b_\ell} b_{\ell-1} \dots b_{\ell-w+1})_2 + b_{\ell-w} \right) \cdot 2^{\ell-w+1} \tag{3.4}$$

which is a function of the $w+1$ digits $b_\ell b_{\ell-1}\dots b_{\ell-w}$.

3.4 Minimality

For an integer n, we define

$$\mathsf{wt}^*(n) := \min\{\mathsf{wt}(\alpha) : \alpha \in D_w{}^*, (\alpha)_2 = n\}.$$

So, $\mathsf{wt}^*(n)$ is the minimum number of nonzero digits required to represent n using a D_w-radix 2 representation. If $\alpha \in D_w{}^*$ and $(\alpha)_2 = n$ then it must be that $\mathsf{wt}(\alpha) \geq \mathsf{wt}^*(n)$; if $\mathsf{wt}(\alpha) = \mathsf{wt}^*(n)$ we say that α has *minimal weight*.

In this section, we will prove the following Theorem:

Theorem 3.7. *Let $w \geq 2$ be an integer. For any $n \in \mathbb{Z}$, the representation returned by $MSF_w(n)$ has a minimal number of nonzero digits.*

It will be convenient to let $\mathrm{MSF}_w(n)$ denote a string returned by the algorithm on input n. To prove Theorem 3.7 we will show that for any $n \in \mathbb{Z}$, $\mathsf{wt}(\mathrm{MSF}_w(n)) = \mathsf{wt}^*(n)$. In doing so, we will make use of a number of short Lemmas concerning the functions $\mathsf{wt}^*(n)$ and $\mathsf{wt}(\mathrm{MSF}_w(n))$.

Lemma 3.8. *If n is even then $\mathsf{wt}^*(n) = \mathsf{wt}^*(n/2)$.*

Proof. Let $(\dots a_2a_1a_0)_2$ be a minimal weight representation of n. Since n is even, $a_0 = 0$ and so $(\dots a_2a_1)_2 = n/2$. Thus, $\mathsf{wt}^*(n/2) \leq \mathsf{wt}^*(n)$. Let $(\dots b_2b_1b_0)_2$ be a minimal weight representation of $n/2$. Then $(\dots b_2b_1b_00)_2 = n$ and so $\mathsf{wt}^*(n) \leq \mathsf{wt}^*(n/2)$. □

Recall the definition of "$n \text{ mods } 2^w$" from Chapter 2, Section 2.2.2. If we write $n = q \cdot 2^w + r$ with $r = n \text{ mods } 2^w$ then $q \cdot 2^w$ is a *multiple of* 2^w *closest* to n. We will make use of this fact later on.

From Chapter 2, we know that the w-NAF of an integer has minimal weight. If n is odd then the least significant digit of its w-NAF is equal to $n \text{ mods } 2^w$. From this fact, we can deduce the following Lemma:

Lemma 3.9. *If n is odd and $r = n \text{ mods } 2^w$, then $\mathsf{wt}^*(n) = 1 + \mathsf{wt}^*((n-r)/2)$.*

Lemma 3.9 can proved in the same way as Lemma 3.8.

To show that $\mathrm{wt}(\mathrm{MSF}_w(n)) = \mathrm{wt}^*(n)$, we will argue by induction on $|n|$. For n odd, it is thus useful to establish that $|(n-r)/2| < |n|$.

Lemma 3.10. *Let n be an odd integer and let $r = n \text{ mods } 2^w$. Then $|(n-r)/2| < |n|$.*

Proof. If $|n| < 2^{w-1}$ then $n \text{ mods } 2^w = n$. Thus,

$$\left|\frac{n-r}{2}\right| = \left|\frac{n-n}{2}\right| = 0 < |n|\,.$$

So we can assume that $|n| \geq 2^{w-1}$. Let q be the integer such that $n = q \cdot 2^w + r$. Since n is nonzero, we have

$$\left|\frac{n-r}{2}\right| < |n| \iff \left|\frac{n-r}{n}\right| < 2 \iff \left|\frac{q}{n}\right| 2^w < 2.$$

Suppose to the contrary that $\left|\frac{q}{n}\right| 2^w \geq 2$. The integers q and n have the same sign, thus

$$\begin{aligned}\left|\frac{q}{n}\right| 2^w \geq 2 &\implies \frac{q}{n}2^w \geq 2\\ &\implies \frac{q}{n}2^w + \frac{r}{n} \geq 2 + \frac{r}{n}\\ &\implies 1 \geq 2 + \frac{r}{n}.\end{aligned} \tag{3.5}$$

Because $r = n \text{ mods } 2^w$, we know that $-2^{w-1} < r \leq 2^{w-1}$. However, n is odd, so r is odd and we have the slightly tighter bound $-2^{w-1} < r < 2^{w-1}$. Since $|n| \geq 2^{w-1}$ we have $1 \geq 2^{w-1}/\,|n|$, thus

$$\begin{aligned}-2^{w-1} < r < 2^{w-1} &\implies \frac{-2^{w-1}}{|n|} < \frac{r}{|n|} < \frac{2^{w-1}}{|n|}\\ &\implies -1 < \frac{r}{|n|} < 1\\ &\implies -1 < \frac{r}{n} < 1.\end{aligned}$$

Since $r/n > -1$, if we continue from (3.5) we arrive at the contradiction $1 > 1$. Thus, $\left|\frac{q}{n}\right| 2^w < 2$ and this is equivalent to the desired result. □

We now give two Lemmas which involve the function $\mathsf{wt}(\mathrm{MSF}_w(n))$.

Lemma 3.11. *If n is an even integer then* $\mathsf{wt}(\mathrm{MSF}_w(n)) = \mathsf{wt}(\mathrm{MSF}_w(n/2))$.

Proof. If $n = 0$ then the result is clearly true, so we can assume $n \neq 0$. Let $\alpha = a_{\ell-1} \dots a_2 a_1 a_0$ be an output of $\mathrm{MSF}_w(n)$ with $a_{\ell-1} \neq 0$. Since n is even and $n = (\alpha)_2$ it must be that $a_0 = 0$. Thus, the strings α and $\alpha' = a_{\ell-1} \dots a_2 a_1$ have the same weight. We show α' is an output of $\mathrm{MSF}_w(n/2)$, and then the result follows from Lemma 3.5.

Let $c = a_{\ell-1} 2^{\ell-1}$; c is an element in $\mathcal{C}_w$ closest to n. Since $a_0 = 0$ and $a_{\ell-1} \neq 0$ it must be that $\ell - 1 \geq 1$, and so c is even. Thus, $c/2 \in \mathcal{C}_w$. Now,

$$c \text{ is closest to } n \implies c/2 \text{ is closest to } n/2,$$

so there is an output of $\mathrm{MSF}_w(n/2)$ where the most significant nonzero digit encodes $c/2 = a_{\ell-1} 2^{\ell-2}$. By repeating this argument, we see that $\alpha' = a_{\ell-1} \dots a_2 a_1$ is indeed an output of $\mathrm{MSF}_w(n/2)$. This proves the result. $\square$

Lemma 3.12. *If c is an element of $\mathcal{C}_w$ closest to n, then* $\mathsf{wt}(\mathrm{MSF}_w(n)) = 1 + \mathsf{wt}(\mathrm{MSF}_w(n - c))$.

Lemma 3.12 follows from the description of Algorithm 3.2.

Now we have everything we need to prove our main result.

Proof of Theorem 3.7. We show that for any $n \in \mathbb{Z}$,

$$\mathsf{wt}(\mathrm{MSF}_w(n)) = \mathsf{wt}^*(n)\,. \tag{3.6}$$

When $n = 0$, $\mathrm{MSF}_w(n)$ returns the empty string; thus

$$\mathsf{wt}(\mathrm{MSF}_w(0)) = 0 = \mathsf{wt}^*(0)\,. \tag{3.7}$$

Also, if n is even then from Lemmas 3.8 and 3.11 we have

$$\mathsf{wt}(\mathrm{MSF}_w(n)) = \mathsf{wt}^*(n) \iff \mathsf{wt}(\mathrm{MSF}_w(n/2)) = \mathsf{wt}^*(n/2)\,. \tag{3.8}$$

Thus, if we can show that (3.6) holds for all n with $|n| \geq 1$ and n odd, then by (3.7) and (3.8), it holds for all n.

Let n be an odd nonzero integer. We argue by induction on $|n|$. For our base cases, we consider the values of n that satisfy $1 \leq |n| < 2^{2w-1}$. We deal with this interval in two parts.

First, suppose $1 \leq |n| < 2^{w-1}$. Then $n \in D_w$ (because n is odd) and thus $\mathsf{wt}(\mathrm{MSF}_w(n)) = 1$. Any odd integer n has $\mathsf{wt}^*(n) \geq 1$, thus we see that

$$\mathsf{wt}(\mathrm{MSF}_w(n)) = 1 = \mathsf{wt}^*(n)\,.$$

Next, suppose $2^{w-1} \leq |n| < 2^{2w-1}$. Note that $\lfloor \lg |n| \rfloor \leq 2w-2$. Let c be an element in $\mathcal{C}_w$ closest to n. Note that c must be even since $|n| \geq 2^{w-1}$. By Lemma 3.3, we have

$$|n-c| \leq 2^{\lfloor \lg |n| \rfloor - w + 1}. \tag{3.9}$$

However,

$$\lfloor \lg |n| \rfloor \leq 2w-2 \implies \lfloor \lg |n| \rfloor - w + 1 \leq w-1,$$

and so

$$|n-c| \leq 2^{w-1}.$$

Since n is odd and c is even, $n-c$ is odd and thus, $n-c \in D_w$. So Algorithm 3.2 uses just two elements of $\mathcal{C}_w$ to represent n (namely, c and $n-c$); thus $\mathsf{wt}(\mathrm{MSF}_w(n)) = 2$. Any odd integer n with $|n| > 2^{w-1}$ (i.e., $n \notin D_w$) has $\mathsf{wt}^*(n) \geq 2$, and from this we see that

$$\mathsf{wt}(\mathrm{MSF}_w(n)) = 2 = \mathsf{wt}^*(n)\,.$$

With our base cases established, we now consider n odd with $|n| \geq 2^{2w-1}$. Note that $\lfloor \lg |n| \rfloor \geq 2w-1$. Let c be an element in $\mathcal{C}_w$ closest to n and let $r = n \text{ mods } 2^w$. We claim that c is also closest to $n-r$. To see this, first note that n lies in one of the intervals

$$[2^{\lfloor \lg |n| \rfloor}, 2^{\lfloor \lg |n| \rfloor + 1}] \quad \text{or} \quad [-2^{\lfloor \lg |n| \rfloor}, -2^{\lfloor \lg |n| \rfloor + 1}].$$

From the proof of Lemma 3.4, we know that all elements of $\mathcal{C}_w$ in these intervals have the form $d \cdot 2^i$ with $d \in D_w$ and

$$i \in \{\lfloor \lg |n| \rfloor - w + 2, \ldots, \lfloor \lg |n| \rfloor, \lfloor \lg |n| \rfloor + 1\}.$$

Thus,

$$i \geq \lfloor \lg |n| \rfloor - w + 2 \geq 2w - 1 - w + 2 = w + 1,$$

and so 2^{w+1} divides c. There are two neighbouring elements of $\mathcal{C}_w$, say c_0 and c_1, such that $n \in [c_0, c_1]$. Let m be the midpoint of $[c_0, c_1]$. We have

$$\begin{aligned}
2^{w+1}|c_0 \text{ and } 2^{w+1}|c_1 &\implies 2^{w+1}|(c_0 + c_1) \\
&\implies 2^{w}|\frac{c_0 + c_1}{2} \\
&\implies 2^{w}|m.
\end{aligned}$$

So c_0, c_1 and m are all multiples of 2^w. One of c_0 or c_1 is equal to c. If $c = c_0$, then $n \in [c, m]$; whereas, if $c = c_1$, then $n \in [m, c]$. In either case, it can be shown that $n - r$ is an element in the same closed interval (this follows because $n - r$ is the multiple of 2^w closest to n). Thus, we see that c is closest to $n - r$. Further, since both c and $n - r$ are even, we have that

$$c/2 \text{ is closest to } (n-r)/2. \tag{3.10}$$

Now we are ready to finish the proof. Notice that, because $2^w|c$, we have

$$n - c \text{ mods } 2^w = n \text{ mods } 2^w = r. \tag{3.11}$$

By induction, we have that $\mathsf{wt}(\mathrm{MSF}_w(n')) = \mathsf{wt}^*(n')$ for all n' with $|n'| <$

$|n|$. Using this and our Lemmas, we find that

$$\begin{aligned}
\mathsf{wt}(\mathrm{MSF}_w(n)) &= 1 + \mathsf{wt}(\mathrm{MSF}_w(n-c)) \qquad \text{(by Lemma 3.12)} \\
&= 1 + \mathsf{wt}^*(n-c) \qquad \text{(by induction)} \\
&= 1 + 1 + \mathsf{wt}^*\left(\frac{(n-c)-r}{2}\right) \qquad \text{(by (3.11) and Lemma 3.9)} \\
&= 1 + 1 + \mathsf{wt}\left(\mathrm{MSF}_w\left(\frac{(n-c)-r}{2}\right)\right) \qquad \text{(by induction)} \\
&= 1 + 1 + \mathsf{wt}\left(\mathrm{MSF}_w\left(\frac{n-r}{2} - \frac{c}{2}\right)\right) \\
&= 1 + \mathsf{wt}\left(\mathrm{MSF}_w\left(\frac{n-r}{2}\right)\right) \qquad \text{(by (3.10) and Lemma 3.12)} \\
&= 1 + \mathsf{wt}^*\left(\frac{n-r}{2}\right) \qquad \text{(by induction)} \\
&= \mathsf{wt}^*(n) \qquad \text{(by Lemma 3.9).}
\end{aligned}$$

Each of the inductive steps above is justified by either Lemma 3.10 or the fact that $|n-c| < |n|$. This concludes the proof. □

3.5 A Characterization of Algorithm 3.2

Let n be a nonzero integer and let $c_1, c_2, \ldots, c_t$ be a sequence of values in $\mathcal{C}_w$ that Algorithm 3.2 selects on input n. By the definition of the algorithm, we have that

$$\begin{array}{lll}
c_1 & \text{is closest to} & n \\
c_2 & \text{is closest to} & n - c_1 \\
c_3 & \text{is closest to} & n - (c_1 + c_2) \\
 & \vdots & \\
c_t & \text{is closest to} & n - (c_1 + c_2 + \cdots + c_{t-1}).
\end{array}$$

Let

$$\begin{aligned} n_1 &= c_1 \\ n_2 &= c_1 + c_2 \\ &\vdots \\ n_t &= c_1 + c_2 + \cdots + c_t. \end{aligned}$$

We can think of the values $n_1, n_2, \ldots, n_t$ as successive approximations to the integer n. Note that $n_t = n$.

Claim 3.13. *For* $1 \le j \le t$, $\mathsf{wt}^*(n_j) = j$.

Proof. Let $(a_{\ell-1} \ldots a_1 a_0)_2$ be the representation of n which results when Algorithm 3.2 selects the values $c_1, c_2, \ldots, c_t$. Note that

$$(a_{\ell-1} \ldots a_1 a_0)_2 = c_1 + c_2 + \cdots + c_t.$$

Because $(a_{\ell-1} \ldots a_1 a_0)_2$ has minimal weight, so too do each of the representations

$$\begin{gathered} (a_{\ell-1} \ldots a_2 a_1 a_0)_2 \\ (a_{\ell-1} \ldots a_2 a_1 0)_2 \\ (a_{\ell-1} \ldots a_2 00)_2 \\ \vdots \\ (a_{\ell-1} 0 \ldots 000)_2. \end{gathered}$$

Note that each representation in this list has length ℓ. For $1 \le j \le t$, a representation in this list has weight j if and only if it stands for the integer n_j. Thus, the claim follows from minimality. $\square$

The approximation n_j has $\mathsf{wt}^*(n_j) = j$. It is natural to consider if, of all the integers that have arithmetic weight j, there might be a better approximation to n than n_j. The following result shows that this is never the case.

Proposition 3.14. *For* $1 \le j \le t$,

$$|n - n_j| \le |n - \widehat{n}_j|$$

for any $\widehat{n}_j \in \mathbb{Z}$ *with* $\mathsf{wt}^*(\widehat{n}_j) = j$.

Proof. Note that $\mathrm{wt}^*(n) = t$. We argue by induction on t. If $t = 1$ or $t = 2$ then it is easy to see that the result is true. Suppose the result fails for some $t > 2$. We show that it also fails for $t - 1$. Let j be an integer in the interval $1 \leq j \leq t$ such that $\left|n - n_j\right| > \left|n - \widehat{n}_j\right|$ for some $\widehat{n}_j \in \mathbb{Z}$ with $\mathrm{wt}^*\left(\widehat{n}_j\right) = j$. By the definition of Algorithm 3.2, we see that $j \neq 1$; and it also must be that $j \neq t$ (because $|n - n_t| = 0$) Thus, $2 \leq j \leq t - 1$.

The integer c_1 is an element in $\mathcal{C}_w$ closest to n; we claim it is also closest to $\widehat{n}_j$. To see this, note that the elements of $\mathcal{C}_w$ partition the number line into intervals. The length of each interval is a power of 2. Consider an interval containing n. One endpoint of this interval must be c_1. Since $\mathrm{wt}^*(n) \neq 1$, the length of this interval cannot be 1, so its length is 2^i for some $i \geq 1$. Let m be the midpoint of this interval (note that m is an integer). The distance between m and c_1 is a power of 2, thus $m - c_1 \in \mathcal{C}_w$, and so $\mathrm{wt}^*(m) = 2$.

Now, n is contained in the half-interval with endpoints c_1 and m. Because $\mathrm{wt}^*(n) \neq 1$, we see that $n \neq c_1$; and because $\mathrm{wt}^*(n) \neq 2$, we see that $n \neq m$. Thus, if

$$|n - c_1| > \left|n - \widehat{n}_j\right|, \text{ and} \tag{3.12}$$

$$|n - m| > \left|n - \widehat{n}_j\right| \tag{3.13}$$

then $\widehat{n}_j$ is in the same half-interval as n, and so c_1 is closest to $\widehat{n}_j$. We show these two inequalities are valid.

We have

$$|n - n_1| > |n - n_2| > |n - n_3| > \cdots > |n - n_t| = 0.$$

Since $\left|n - n_j\right| > \left|n - \widehat{n}_j\right|$, with $j \geq 2$, we see that

$$\begin{aligned} |n - n_1| &> \left|n - \widehat{n}_j\right|, \text{ and} \\ |n - n_2| &> \left|n - \widehat{n}_j\right| \end{aligned}$$

By noting that $n_1 = c_1$, we get (3.12). To get (3.13), first recall that $m - c_1 \in \mathcal{C}_w$. By definition of Algorithm 3.2, we see that

$$\begin{aligned} &|(n - c_1) - c_2| \leq |(n - c_1) - (m - c_1)| \\ \Longrightarrow\ &|n - (c_1 + c_2)| \leq |n - m|. \end{aligned}$$

By noting that $n_2 = c_1 + c_2$, we get (3.13). So, c_1 is indeed closest to $\widehat{n}_j$.

Now,

$$\begin{aligned} & \left|n - n_j\right| > \left|n - \widehat{n}_j\right| \\ \Longrightarrow\ & \left|(n - c_1) - (n_j - c_1)\right| > \left|(n - c_1) - (\widehat{n}_j - c_1)\right|, \end{aligned} \tag{3.14}$$

and, because c_1 is closest to $\widehat{n}_j$, we see that

$$\mathsf{wt}^*\left(\widehat{n}_j - c_1\right) = \mathsf{wt}^*\left(\widehat{n}_j\right) - 1 = j - 1.$$

On input $n - c_1$, Algorithm 3.2 can create a representation of $n - c_1$ by selecting the values $c_2, c_3, \ldots, c_t$. Note that $\mathsf{wt}^*(n - c_1) = t - 1$. These values result in $t - 1$ approximations to $n - c_1$. Approximation $j - 1$ is equal to $c_2 + c_3 + \cdots + c_j = n_j - c_1$, however, by (3.14), we see that this approximation is not optimal. Thus, the result fails for $t - 1$. This concludes the induction proof. □

We have shown that any output of Algorithm 3.2 satisfies Proposition 3.14. The converse of this statement is also true.

Let $(a_{\ell-1} \ldots a_1 a_0)_2$ be a representation of a nonzero integer n which contains exactly t nonzero digits. Decompose this representation into a sum of t representations, each of which contains exactly one of the nonzero digits. Order the representations in this sum by length, longest to shortest. Reading the representations as integers, let $c_1, c_2, \ldots, c_t$ be the terms of this ordered sum. For example, for the representation $(3100\overline{1})_2$, we write

$$\begin{aligned} (3100\overline{1})_2 &= (30000)_2 + (01000)_2 + (0000\overline{1})_2 \\ &= 3 \cdot 2^4 + 1 \cdot 2^3 - 1 \cdot 2^0, \end{aligned}$$

and so $c_1 = 48, c_2 = 8, c_3 = -1$. As before, we let

$$\begin{aligned} n_1 &= c_1 \\ n_2 &= c_1 + c_2 \\ &\vdots \\ n_t &= c_1 + c_2 + \cdots + c_t. \end{aligned}$$

Note that $n_t = n$. We have the following result:

Proposition 3.15. *If, for $1 \le j \le t$,*

$$\left| n - n_j \right| \le \left| n - \widehat{n}_j \right|$$

for any $\widehat{n}_j \in \mathbb{Z}$ with $\mathrm{wt}^*\left(\widehat{n}_j\right) = j$, then $(a_{\ell-1} \ldots a_1 a_0)_2$ is an output of Algorithm 3.2 on input n.

Proof. We must show that, for $1 \le j \le t$,

$$c_j \text{ is closest to } n - (c_1 + c_2 + \cdots + c_{j-1}),$$

which is equivalent to showing that, for any $c \in \mathcal{C}_w$,

$$\left| (n - c_1 - c_2 - \cdots - c_{j-1}) - c_j \right| \le \left| (n - c_1 - c_2 - \cdots - c_{j-1}) - c \right| . \quad (3.15)$$

If we take $j = 1$ in our hypothesis, we have

$$|n - n_1| \le |n - \widehat{n}_1|$$

for any $\widehat{n}_1 \in \mathbb{Z}$ with $\mathrm{wt}^*(\widehat{n}_1) = 1$; that is, for any $\widehat{n}_1 = c$ where $c \in \mathcal{C}_w$. Since $n_1 = c_1$, we see that (3.15) holds for $j = 1$.

Now, take any j with $2 \le j \le t$. For any $c \in \mathcal{C}_w$, let $\widehat{n}_j = c_1 + c_2 + \cdots + c_{j-1} + c$. Note that $\mathrm{wt}^*\left(\widehat{n}_j\right) \le j$. If $\mathrm{wt}^*\left(\widehat{n}_j\right) = j$, then by hypothesis

$$\begin{aligned} \left| n - n_j \right| &\le \left| n - \widehat{n}_j \right| \\ \Longrightarrow \left| n - (c_1 + c_2 + \cdots + c_j) \right| &\le \left| n - (c_1 + c_2 + \cdots + c_{j-1} + c) \right| , \end{aligned}$$

and this gives (3.15). Suppose $\mathrm{wt}^*\left(\widehat{n}_j\right) = j_0$ where $j_0 < j$. By hypothesis, we have

$$\left| n - n_{j_0} \right| \le \left| n - \widehat{n}_j \right| .$$

If we can show that $\left| n - n_j \right| \le \left| n - n_{j_0} \right|$, then (3.15) follows as in the previous case. To deduce that $\left| n - n_j \right| \le \left| n - n_{j_0} \right|$, we apply the following claim.

Claim 3.16. *Let n be a nonzero integer with* $\mathsf{wt}^*(n) = t$ *and let j be an integer with* $1 \leq j \leq t$. *If, of all integers with arithmetic weight equal to j, n' is closest to n, then n' is also closest to n of all integers with arithmetic weight* at most *j.*

The claim can be proved using the same method of induction employed in the proof of Proposition 3.14. Before we can apply the claim, we first show that $\mathsf{wt}^*(n) = t$. Since $(a_{\ell-1} \ldots a_1 a_0)_2 = n$ and this representation has t nonzero digits, we have $\mathsf{wt}^*(n) \leq t$. If $\mathsf{wt}^*(n) < t$, then, for $j = \mathsf{wt}^*(n)$, we can take $\widehat{n}_j = n$, and then

$$0 < |n - n_j| \leq |n - n| = 0$$

which is a contradiction. So, $\mathsf{wt}^*(n) = t$. Because we now know that $(a_{\ell-1} \ldots a_1 a_0)_2$ is a minimal weight representation, we see that $\mathsf{wt}^*(n_j) = j$ for $1 \leq j \leq t$.

By hypothesis, we have that n_j is the integer with arithmetic weight j closest to n. Now, since $j_0 < j$, we have $\mathsf{wt}^*(n_{j_0}) < j$. Thus, if we apply Claim 3.16, we have that $|n - n_j| \leq |n - n_{j_0}|$, and the result follows. □

3.6 Related Work

Avanzi [1] independently obtained similar results which were presented at SAC 2004. In particular, Avanzi describes a deterministic algorithm which constructs D_w-radix 2 representations by scanning the binary representation of an integer from left to right. He also proves that these representation have minimal weight. As the input is scanned, Avanzi's algorithm works by applying arithmetic operations to windows of $w+1$ digits; his algorithm does not require a stored table. By comparing the expression for c in equation (3.4) to Avanzi's algorithm, it can be shown that Avanzi's algorithm is a deterministic implementation of Algorithm 3.2; that is, Avanzi's algorithm works by choosing closest elements from the set $\mathcal{C}_w$.

At CRYPTO 2004, Okeya, Schmidt-Samoa, Spahn and Takagi [36] presented a very simple technique that allows D_w-radix 2 representations to be constructed from either right to left or left to right. They consider a canonical $\{0, \pm 1\}$-radix 2 representation of an integer, n, constructed by the digit-wise subtraction of the binary representation of n from that of $2n$. Once this representation is constructed, D_w-radix 2 representations can be obtained by sliding windows of width-w across it. Sliding the window right to left gives the w-NAF, and sliding the window left to right gives the same representation constructed by Avanzi's algorithm. Okeya et al. show that the average density of nonzero digits in their left-to-right representations is asymptotically $1/(w+1)$, however they do not prove minimality.

The same canonical $\{0, \pm 1\}$-radix 2 representation defined by Okeya et al. can be found in work by Grabner, Heuberger, Prodinger and Thuswaldner [13]. Grabner et al. use these representations to construct minimal weight joint $\{0, \pm 1\}$-radix 2 representations of pairs of integers from left to right. Heuberger, Katti, Prodinger and Ruan [16] also use the canonical representations. They generalize the results of Grabner et al. to joint representations of $d \geq 2$ integers. As well, Heuberger et al. show how Avanzi's left-to-right algorithm can be obtained from the canonical representation.

What is unique to our work is the idea of choosing closest elements in the set $\mathcal{C}_w$, our simple nondeterministic algorithm, our technique for proving minimality and our method of incorporating our left-to-right representations into the algorithm for scalar multiplication.

3.7 Remarks

In proving that our new representations have a minimal number of nonzero digits, we essentially dealt with the following two statements concerning odd integers:

$$\mathrm{wt}^*(n) = 1 + \mathrm{wt}^*((n-r)/2) \quad \text{where } r = n \text{ mods } 2^w \tag{3.16}$$

$$\mathrm{wt}^*(n) = 1 + \mathrm{wt}^*(n-c) \quad \text{where } c \in \mathcal{C}_w \text{ is closest to } n. \tag{3.17}$$

In our proof, we noted that (3.16) is true (by the minimality of the w-NAF) and then showed that (3.16) implies (3.17). The same arguments can be used to show that (3.17) implies (3.16). Thus, (3.16) and (3.17) are logically equivalent, which is perhaps surprising.

The w-NAF has a very simple combinatorial description: they are the representations which use the digits D_w and have the property that of every w consecutive digits, at most one is nonzero. From this description, it is very easy to look at a representation and quickly decide whether or not it is a w-NAF. For our new representations, this does not appear to be quite so easy.

Chapter 4

Joint Representations

4.1 Introduction

Let $r \geq 1$ be an integer and let $D \subset \mathbb{Z}$ be a set of digits with $0 \in D$. A *D-radix 2 joint representation* is a finite sum of the form

$$\sum_{i \geq 0} A_i 2^i, \quad A_i \in D^{r \times 1}.$$

This sum evaluates to a vector $N \in \mathbb{Z}^{r \times 1}$. We use the following notation to denote such sums:

$$(\ldots A_2 A_1 A_0)_2 = \cdots + A_2 2^2 + A_1 2^1 + A_0,$$

or,

$$\begin{pmatrix} \ldots a_{12} a_{11} a_{10} \\ \ldots a_{22} a_{21} a_{20} \\ \vdots \\ \ldots a_{r2} a_{r1} a_{r0} \end{pmatrix}_2 = \cdots + \begin{pmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{r2} \end{pmatrix} 2^2 + \begin{pmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{r1} \end{pmatrix} 2^1 + \begin{pmatrix} a_{10} \\ a_{20} \\ \vdots \\ a_{r0} \end{pmatrix} 2^0,$$

where each $a_{ji} \in D$. The column vectors A_i are defined for each $i \geq 0$. Since the sum is finite, this means that all but a finite number of the A_i's are zero vectors.

Example 4.1. Here is a $\{0,1\}$-radix 2 joint representation:

$$\begin{pmatrix} 0111 \\ 1011 \\ 1101 \\ 1110 \end{pmatrix}_2 = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} 2^3 + \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} 2^2 + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} 2^1 + \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} 2^0 = \begin{pmatrix} 7 \\ 11 \\ 13 \\ 14 \end{pmatrix}. \qquad \Diamond$$

Let $\mathcal{A} = \ldots A_2 A_1 A_0$ be a finite length string of column vectors from $D^{r\times 1}$ (i.e., $\mathcal{A} \in (D^{r\times 1})^*$). Then $(\mathcal{A})_2$ is an r-row joint representation. We denote the number of nonzero columns in $(\mathcal{A})_2$ by $\mathsf{wt}(\mathcal{A})$. This value is often referred to as the *joint weight*, or simply, the *weight*, of $(\mathcal{A})_2$. We define the length of $(\mathcal{A})_2$ as follows:

$$\mathsf{length}(\mathcal{A}) := \min\{\ell \in \mathbb{Z} : \ell \geq 0, \text{ and for any } i \geq \ell,\ A_i \text{ is a zero column}\}.$$

Note that if $\ell = \mathsf{length}(\mathcal{A})$ and $\ell > 0$, then $A_{\ell-1}$ is the leftmost nonzero column of $(\mathcal{A})_2$.

Example 4.2. In the previous example, the representation has weight 4 and length 4. It is easy to see that this is the only joint representation of the vector $(7, 11, 13, 14)^T$ which uses the digits $\{0,1\}$. If we instead use the digits $\{0,1,2,3\}$, then $(7,11,13,14)^T$ has a number of different representations, one of which is

$$\begin{pmatrix} 103 \\ 203 \\ 301 \\ 302 \end{pmatrix}_2 = \begin{pmatrix} 7 \\ 11 \\ 13 \\ 14 \end{pmatrix}.$$

This representation has length 3 and weight 2. $\Diamond$

In this chapter, we consider the problem of constructing *minimal weight* r-row joint representations. Perhaps the most important observation we make is that minimal weight representations and *colexicographically* smallest representations share some of the same properties. Once we demonstrate this commonality, then, for a given set of digits, it is natural to ask whether a colexicographically smallest representation has minimal weight. We show that, for certain families of digit sets, this is indeed true.

Notice that joint representations generalize the radix 2 integer representations that we considered in previous chapters. It is often instructive to specialize a result on r-row joint representations to the case $r = 1$ and compare this to what we have learned so far. When we do this, we see that many of the canonical integer representations that have been proposed in the literature can be interpreted as colexicographically smallest representations.

4.2 Solinas' Problems

Solinas [42] introduced the study of joint representations in his treatment of the following problem:

Problem 4.3. *Given a vector* $N \in \mathbb{Z}^{2\times 1}$*, construct a minimal weight joint representation of* N *using the digits* $\{0, \pm 1\}$.

This problem arose when Solinas considered how to optimize Straus' method [43] for computing a linear combination of two elliptic curve points.

Straus' method computes $n_1P + n_2Q$ using doubling, addition and subtraction operations as it processes a $\{0, \pm 1\}$-joint representation of $N = (n_1, n_2)^T$ from left to right, one column at a time. Any $\{0, \pm 1\}$-joint representation of N can be used. However, since an addition/subtraction operation is required for each nonzero column in the representation of N, it is advantageous to use a minimal weight representation.

Solinas solved Problem 4.3 by describing an algorithm which constructs a canonical joint representation for pairs of integers called the *joint sparse form* (JSF). Solinas developed the JSF as a generalization of the NAF. Recall that the NAF has the following properties:

1. every integer has at most one NAF (uniqueness),
2. every integer has a NAF (existence),
3. the NAF can be efficiently computed (efficiency),

4. the NAF has minimal weight (minimality).

The JSF satisfies analogous properties:

1. every pair of integers has at most one JSF (uniqueness),
2. every pair of integers has a JSF (existence),
3. the JSF can be efficiently computed (efficiency),
4. the JSF has minimal weight (minimality).

By property 4, we see that the JSF is an optimal representation to use with Straus' method. Properties 2 and 3 answer some of the practical questions we might encounter in employing the JSF; namely, that this representation can always be efficiently constructed. Perhaps the least important property of the JSF, with respect to its use with Straus' method, is uniqueness.

Solinas also posed three additional problems for further research.

Problem 4.4. *Give an r-row analogue of the JSF.*

This problem was solved independently by Proos [38] and by Grabner, Heuberger and Prodinger [12]. In fact, before Solinas had posed this problem, Proos had worked on the related problem of reducing the number of nonzero columns in a "Lim-Lee" [35] combing table[1] using the digits $\{0, \pm 1\}$. Grabner, Heuberger and Prodinger's solution is particularly lucid and their proof of minimality is quite elegant.

Solinas' next problem involves a familiar topic. The JSF is constructed by sliding a window from right to left across the $\{0, 1\}$-joint representation of a vector $N \in \mathbb{Z}^{2\times 1}$. However, Straus' algorithm works left-to-right. This suggests the following problem:

Problem 4.5. *Find an analogue of the JSF that can be built using a left-to-right method.*

[1]Bernstein [4] points out that the term "Lim-Lee combining table" is controversial since this method had previously been published by Pippenger [37]

Katti [19] briefly describes a left-to-right algorithm for computing a $\{0, \pm 1\}$-joint representation of a vector $N \in \mathbb{Z}^{2\times 1}$. No analysis is given and no claims are made about minimality. A spurious example is given that "shows" that the output of this left-to-right algorithm can have fewer nonzero columns than the corresponding JSF (thereby contradicting the minimality of the JSF). The problem with the example is that the representation of $N = (53, 102)^T$ that is used is not a JSF; Katti appears to have copied a typographic error from one of Solinas' examples (page 6 of [42]). Grabner, Heuberger, Prodinger and Thuswaldner [13] independently discovered Katti's algorithm and were able to prove minimality using an argument based on generating functions. Katti and Ruan [20] also offer a proof. At a later date, Heuberger, Katti, Prodinger and Ruan [16] show how to extend their algorithm to an arbitrary number of rows and they also prove minimality.

Solinas' final problem, which has received little attention, is the following:

Problem 4.6. *Give an analogue of the JSF which uses digits other than* $\{0, \pm 1\}$ *(e.g.,* $\{0, 1, 3\}$ *or* $\{0, \pm 1, \pm 3\}$*).*

Most of the results presented in this chapter are concerned with this problem. Solinas suggested the digits $\{0, 1, 3\}$ because every nonnegative integer has a unique $\{0, 1, 3\}$-nonadjacent form that has minimal weight. A proof of this fact does not seem to appear in the literature; however, it is not difficult to construct one by modifying the argument used in Chapter 2 to prove the minimality of the w-NAF (alternatively, this fact can be deduced from Theorem 4.15). We know also that every integer has a unique 3-NAF which uses the digits $\{0, \pm 1, \pm 3\}$.

The minimality of the w-NAF was proven independently by Avanzi [1]. Avanzi makes the important observation that the restriction on the set of w-NAF digits may be relaxed to include both even and odd digits. For example, the 3-NAF of an integer has minimal weight amongst all radix 2 representations which use the digits $\{0, \pm 1, \pm 3\}$. However, it is also true that the 3-NAF has minimal weight amongst all radix 2 representations which use the digits $\{0, \pm 1, \pm 2, \pm 3\}$. The same is true for the $\{0, 1, 3\}$-nonadjacent form; it has minimal weight amongst all radix 2 representations which use the digits $\{0, 1, 2, 3\}$. These facts suggest that it might be more natural to look for an analogue of the JSF which uses digits $\{0, \pm 1, \pm 2, \pm 3\}$ or $\{0, 1, 2, 3\}$.

4.3 Colexicographic Order

Let $r \geq 1$ be an integer and let $D \subset \mathbb{Z}$ be a finite set of digits with $0 \in D$. For any $N \in \mathbb{Z}^{r \times 1}$, consider the set of all r-row joint representations of N which use the digits D. We can impose an order on the representations in this set using the approach from Chapter 2.

Each joint representation of N identifies a unique finite length column-string in $(D^{r \times 1})^*$ with no leading zero columns. If $(\mathcal{A})_2 = N$ and $\mathcal{A} \in (D^{r \times 1})^*$, then we can assume, without loss of generality, that $\mathcal{A}$ has no leading zero columns. From the column-string $\mathcal{A}$, we derive a binary string $\mathsf{char}(\mathcal{A})$. The string $\mathsf{char}(\mathcal{A})$ is defined as follows: if $\mathcal{A} = \ldots A_2 A_1 A_0$ then $\mathsf{char}(\mathcal{A}) = \ldots a_2 a_1 a_0$ where

$$a_i := \begin{cases} 0 & \text{if } A_i \text{ is a zero column} \\ 1 & \text{otherwise.} \end{cases}$$

Now, for any two joint representations, $(\mathcal{A})_2$, $(\mathcal{B})_2$, of N, we write $\mathcal{A} \preceq \mathcal{B}$ if $\mathsf{char}(\mathcal{A})$ is less than or equal to $\mathsf{char}(\mathcal{B})$ when they are compared colexicographically.

A colexicographically *smallest* representation of a vector N has a recursive property, which is described in the following lemma:

Lemma 4.7. *Let $r \geq 1$ be an integer. If $(\ldots A_3 A_2 A_1 A_0)_2$ is a colexicographicallly smallest representation of a vector $N \in \mathbb{Z}^{r \times 1}$, then $(\ldots A_3 A_2 A_1)_2$ is a colexicographically smallest representation of $(N - A_0)/2$.*

Proof. Let $\mathcal{A} = \ldots A_3 A_2 A_1 A_0$ and $\mathcal{A}' = \ldots A_3 A_2 A_1$. Let $(\mathcal{B}')_2$ be a colexicographically smallest representation of $(N - A_0)/2$. Suppose $\mathsf{char}(\mathcal{B}')$ is strictly less than $\mathsf{char}(\mathcal{A}')$, colexicographically. Then

$$\begin{aligned} & \mathcal{B}' \prec \mathcal{A}' \\ \Longrightarrow\ & \mathcal{B}' \| A_0 \prec \mathcal{A}' \| A_0 \\ \Longrightarrow\ & \mathcal{B}' \| A_0 \prec \mathcal{A}. \end{aligned}$$

Since $(\mathcal{B}' \| A_0)_2 = N$, we see that $(\mathcal{A})_2$ is not a colexicographically smallest representation of N. □

The same recursive property is true of minimal weight joint representations.

Lemma 4.8. *Let $r \geq 1$ be an integer. If $(\ldots A_3 A_2 A_1 A_0)_2$ is a minimal weight r-row joint representation of a vector $N \in \mathbb{Z}^{r \times 1}$, then $(\ldots A_3 A_2 A_1)_2$ is a minimal weight representation of $(N - A_0)/2$.*

Proof. Let $\mathcal{A} = \ldots A_3 A_2 A_1 A_0$ and $\mathcal{A}' = \ldots A_3 A_2 A_1$. Suppose $(\mathcal{B}')_2$ is a representation of $(N - A_0)/2$ that has fewer nonzero columns than $(\mathcal{A}')_2$. Then

$$\begin{aligned} & \mathsf{wt}\big(\mathcal{B}'\big) < \mathsf{wt}\big(\mathcal{A}'\big) \\ \Longrightarrow\ & \mathsf{wt}\big(\mathcal{B}' \| A_0\big) < \mathsf{wt}\big(\mathcal{A}' \| A_0\big) \\ \Longrightarrow\ & \mathsf{wt}\big(\mathcal{B}' \| A_0\big) < \mathsf{wt}(\mathcal{A}). \end{aligned}$$

Since $(\mathcal{B}' \| A_0)_2 = N$, we see that $\mathcal{A}$ is not a minimal weight representation of N. □

Notice that these lemmas are true for *any* digit set $D \subset \mathbb{Z}$. We will describe other commonalities between colexicographically smallest representations and minimal weight representations in the following sections.

4.4 Unsigned Digits

For any integer $w \geq 2$, we define the set of digits

$$E_w := \{n \bmod 2^w : n \in \mathbb{Z}\} = \{0, 1, 2, \ldots, 2^w - 1\}.$$

Notice that E_w is quite different from the set of w-NAF digits that we considered in Chapters 2 and 3. The set E_w contains only nonnegative digits. As well, E_w contains both even and odd nonzero integers. Note that $|D_w| = 2^{w-1} + 1$, while $|E_w| = 2^w$.

If we consider joint representations which use the digits E_w then we can observe additional similarities between colexicographically smallest representations and minimal weight representations.

Lemma 4.9. *Let $r \geq 1$ be an integer. If $(\mathcal{A})_2$ is a colexicographically smallest representation of a vector $N \in \mathbb{Z}^{r\times 1}$ which uses the digits E_w, then every nonzero column of $(\mathcal{A})_2$ must contain an odd digit.*

Proof. Let $\mathcal{A} = \ldots A_2 A_1 A_0$ and suppose $\mathcal{A}$ contains a nonzero column consisting of only even digits. By Lemma 4.7, we can assume that A_0 is such a nonzero column. This implies that N is composed of only even integers. Consider the $\{0, 1\}$-joint representation of N. The least significant column of this representation must be a zero column. However, this representation contradicts the fact that $(\mathcal{A})_2$ is a colexicographically smallest representation of N because the least significant column of $(\mathcal{A})_2$ is nonzero. □

Lemma 4.10. *Let $r \geq 1$ be an integer. Any $r \times 1$ vector N of nonnegative integers has a minimal weight E_w-joint representation where each nonzero column contains an odd digit.*

Proof. Suppose $(\mathcal{A})_2$ is a minimal weight representation of N that has a nonzero column consisting of only even digits. Let $\mathcal{A} = \ldots A_2 A_1 A_0$. By Lemma 4.8, we can assume that A_0 is such a nonzero column. We describe how to modify $(\mathcal{A})_2$ so that A_0 becomes a zero column while the joint weight does not change. Once we establish this, the result follows from Lemma 4.8.

Let A_t be the first zero column that follows A_0:

$$\mathcal{A} = \ldots A_t \underbrace{A_{t-1} \ldots A_1 A_0}_{\text{nonzero}}.$$

Row j of $(\mathcal{A})_2$ is a E_w-radix 2 representation, $(\ldots a_{j2} a_{j1} a_{j0})_2$. Note that $a_{j\,t} = 0$ and a_{j0} is even. We show that there is a representation $(b_{j\,t} b_{j\,t-1} \ldots b_{j1} 0)_2$ with

$$(b_{j\,t} b_{j\,t-1} \ldots b_{j1} 0)_2 = (0 a_{j\,t-1} \ldots a_{j1} a_{j0})_2.$$

If $a_{j0} = 0$, there is nothing to prove, so we can assume a_{j0} is nonzero. Observe that

$$\begin{aligned}(0 a_{j\,t-1} \ldots a_{j1} a_{j0})_2 &= (0 a_{j\,t-1} \ldots a_{j1} 0)_2 + a_{j0} \\ &= 2\left((0 a_{j\,t-1} \ldots a_{j1})_2 + a_{j0}/2\right).\end{aligned}$$

Consider the sum $(0 a_{j\,t-1} \ldots a_{j1})_2 + a_{j0}/2$. Let $d = 2^w - 1$. Note that d is the largest digit in E_w. Now,

$$\begin{aligned}0 < (0 a_{j\,t-1} \ldots a_{j1})_2 + a_{j0}/2 &\le (0 \underbrace{d \ldots d}_{t-1})_2 + a_{j0}/2 \\ &< (0 \underbrace{d \ldots d}_{t-1})_2 + d \\ &= (d \underbrace{0 \ldots 0}_{t-1})_2.\end{aligned}$$

From this bound, we can conclude that the integer $(0 a_{j\,t-1} \ldots a_{j1})_2 + a_{j0}/2$ has a E_w-radix 2 representation of length at most t. Let $(b_{j\,t} b_{j\,t-1} \ldots b_{j1})_2$ be such a representation. Then,

$$\begin{aligned}& (b_{j\,t} b_{j\,t-1} \ldots b_{j1})_2 = (0 a_{j\,t-1} \ldots a_{j1})_2 + a_{j0}/2 \\ \implies\ & (b_{j\,t} b_{j\,t-1} \ldots b_{j1} 0)_2 = (0 a_{j\,t-1} \ldots a_{j1} a_{j0})_2.\end{aligned}$$

So, we see that we can replace the $t+1$ least significant digits in row j of $(\mathcal{A})_2$, with $b_{j\,t} b_{j\,t-1} \ldots b_{j1} 0$. We do this for every row of $(\mathcal{A})_2$ that contains a nonzero digit in column A_0. These replacements affect only the the $t+1$ least significant columns of $(\mathcal{A})_2$.

This results in a new representation of N; call it $(\mathcal{A}')_2$. Let $\mathcal{A}' = \ldots A'_2 A'_1 A'_0$. Note that A'_0 is a zero column. Since $(\mathcal{A})_2$ has minimal weight it must be that A'_t is nonzero; otherwise, $\mathsf{wt}(\mathcal{A}') < \mathsf{wt}(\mathcal{A})$ which contradicts the fact that $(\mathcal{A})_2$ has minimal weight. Thus, $\mathsf{wt}(\mathcal{A}') = \mathsf{wt}(\mathcal{A})$ and so $(\mathcal{A}')_2$ has minimal weight. This proves the lemma. $\square$

Example 4.11. The column transformations described in the previous proof can be applied to *any* joint representation (not necessarily one that has minimal weight) which contains a nonzero column consisting of only even digits. We apply the transformations to the following joint representation of $N = (18, 24)^T$ which uses the digits $E_3 = \{0, 1, 2, 3, 4, 5, 6, 7\}$:

$$\begin{pmatrix} 18 \\ 24 \end{pmatrix} = \begin{pmatrix} 023\boxed{4} \\ 017\boxed{6} \end{pmatrix}_2 = \begin{pmatrix} 0\boxed{2}50 \\ 0\boxed{6}00 \end{pmatrix}_2 = \begin{pmatrix} 1050 \\ 3000 \end{pmatrix}_2 .$$

Notice that the joint weight of the representation decreased when we applied one of the transformations. This could not have happened if we started out with a minimal weight representation of $(18, 24)^T$. $\lozenge$

There are also some differences between minimal weight and colexicographically smallest representations. We will see that every $r \times 1$ vector N of nonnegative integers has exactly one colexicographically smallest representation in the digits E_w, however this vector may have many different minimal weight representations.

Example 4.12. For any $w \geq 2$, it is easy to find a vector N that has more than one minimal weight E_w-joint representation. For example,

$$\begin{pmatrix} 128 \\ 128 \end{pmatrix} = \begin{pmatrix} 10000000 \\ 10000000 \end{pmatrix}_2 = \begin{pmatrix} 02000000 \\ 02000000 \end{pmatrix}_2 . \qquad \lozenge$$

Lemma 4.13. *Every $r \times 1$ vector N of nonnegative integers has a unique colexicographically smallest representation which uses the digits E_w.*

Proof. Since $\{0, 1\} \subset E_w$, every vector N of nonnegative integers has a E_w-joint representation since we can just take each row to be a $\{0, 1\}$-radix 2 representation. Thus, every vector N has a colexicographically smallest representation; what we need to show is that it cannot have more than one such representation.

The zero vector has exactly one E_w-joint representation, so we can assume that N is not the zero vector. Let $(\mathcal{A})_2$ and $(\mathcal{B})_2$ be colexicographically smallest joint representations of N. We suppose $\mathcal{A} \neq \mathcal{B}$ and then show that this leads to a contradiction.

Let ℓ and ℓ' be the respective lengths of $\mathcal{A}$ and $\mathcal{B}$. Write

$$\mathcal{A} = A_{\ell-1} \dots A_2 A_1 A_0 \quad \text{and} \quad \mathcal{B} = B_{\ell'-1} \dots B_2 B_1 B_0.$$

We can assume that ℓ is as small as possible (i.e., there is no other choice of N which leads to a shorter $\mathcal{A}$).

Let

$$\mathcal{A}' = A_{\ell-1} \dots A_2 A_1 \quad \text{and} \quad \mathcal{B}' = B_{\ell'-1} \dots B_2 B_1.$$

If $A_0 = B_0$, then $(\mathcal{A}')_2 = (\mathcal{B}')_2$. So, by Lemma 4.7, $(N - A_0)/2$ has two different colexicographically smallest representations, contrary to the minimality of ℓ. Thus $A_0 \neq B_0$.

Neither of A_0 or B_0 can be a zero column. For, if B_0 is a zero column, then since $\mathcal{A} \preceq \mathcal{B}$, it must be that A_0 is a zero column. But this contradicts the fact that $A_0 \neq B_0$. So, B_0 is a nonzero column and the same argument shows that A_0 is a nonzero column.

By Lemma 4.9, one of the coordinates of N is an odd integer. Consider the $\{0, 1\}$-joint representation of N. The w least significant columns of this representation can be transformed by replacing the contents of each row with a digit from E_w in the least significant column. This results in a E_w-joint representation where the least significant column is nonzero and each of the $w - 1$ columns to the left of it are zero columns. This representation cannot be colexicographically smaller than $\mathcal{A}$ or $\mathcal{B}$, so it must be that

$$\mathcal{A} = A_{\ell-1} \dots \underbrace{A_{w-1} \dots A_2 A_1}_{\text{zero columns}} A_0, \quad \text{and} \quad \mathcal{B} = B_{\ell'-1} \dots \underbrace{B_{w-1} \dots B_2 B_1}_{\text{zero columns}} B_0;$$

otherwise, $\mathcal{A}$ and $\mathcal{B}$ would not be colexicographically smallest representations. Now, from the value of $N \bmod 2^w$, we can conclude that $A_0 = B_0$; however, this contradicts the minimality of ℓ.

Thus, it must be that $\mathcal{A} = \mathcal{B}$; that is, there is exactly one colexicographically smallest representation of N. □

The preceding proof also tells us how to build the colexicographically smallest representation of a vector of nonnegative integers. We start with the $\{0, 1\}$-joint representation of N and slide a w-column window from right to left across it. Each time the rightmost column of the window contains a nonzero column, we replace the contents of each row in the window with a digit from E_w and advance the window to the left. This process can also be phrased in terms of integer operations:

```
**Algorithm 4.14:** COLEXI-SMALLEST-$E_w$-JOINT-REP$(N, w)$

comment: $N$ is an $r \times 1$ vector of nonnegative integers.
$\mathcal{A} \leftarrow \epsilon$
while $N \neq \vec{0}$
  do
    if $N \bmod 2 \neq \vec{0}$
      then $A_i \leftarrow N \bmod 2^w$
      else $A_i \leftarrow \vec{0}$
    $\mathcal{A} \leftarrow A_i \parallel \mathcal{A}$
    $N \leftarrow (N - A_i)/2$
return $\mathcal{A}$
```

This algorithm is a slight modification of the one Solinas [41] gives for the w-NAF (although, here we are using *unsigned* digits). Note that if we take $r = 1$ and $w = 2$, then the algorithm will construct the $\{0, 1, 3\}$-nonadjacent form of a nonnegative integer.

4.4.1 Minimality of Colexi Smallest Representations

In this section we prove the following theorem:

Theorem 4.15. *Let $r \geq 1$ and $w \geq 2$ be integers. Then, for any $r \times 1$ vector of nonnegative integers N, the colexicographically smallest E_w-joint representation of N has minimal weight.*

Proof. The proof is essentially the same as the proof that the w-NAF has minimal weight. The result is clearly true when N is the zero vector, so we can assume that N is nonzero.

Let $(\mathcal{A})_2$ be the colexicographically smallest representation of N. Suppose $(\mathcal{A})_2$ does not have minimal weight. Then there is a minimal weight representation $(\mathcal{B})_2 = N$ with $\mathsf{wt}(\mathcal{B}) < \mathsf{wt}(\mathcal{A})$. By Lemma 4.10, we can assume that every nonzero column of $(\mathcal{B})_2$ contains an odd digit.

Let ℓ and ℓ' be the respective lengths of $\mathcal{A}$ and $\mathcal{B}$. Write

$$\mathcal{A} = A_{\ell-1} \ldots A_2 A_1 A_0, \quad \text{and} \quad \mathcal{B} = B_{\ell'-1} \ldots B_2 B_1 B_0.$$

We can assume that ℓ is as small as possible (i.e., there is no other choice of N which gives a shorter $\mathcal{A}$).

Since ℓ is minimal, it must be that $A_0 \neq B_0$. If B_0 is a zero column, then, because $\mathcal{A} \preceq \mathcal{B}$, A_0 must also be a zero column; however, this contradicts the fact that $A_0 \neq B_0$. So, B_0 is a nonzero column. Since B_0 contains an odd digit, some coordinate of N must be odd. This implies that A_0 is nonzero.

Let

$$\mathcal{A}' = A_{\ell-1} \ldots A_{w+1} A_w, \quad \text{and} \quad \mathcal{B}' = B_{\ell'-1} \ldots B_{w+1} B_w.$$

Because $(\mathcal{A})_2$ is the colexicographically smallest representation of N and A_0 is nonzero, all of $A_{w-1} \ldots A_2 A_1$ must be zero columns. So,

$$\mathsf{wt}(\mathcal{A}) = \mathsf{wt}\big(\mathcal{A}'\big) + 1.$$

Also, if B_0 is the only nonzero column of $B_{w-1} \ldots B_1 B_0$, then from the value of $N \bmod 2^w$ we could deduce that $A_0 = B_0$. However, since $A_0 \neq B_0$, it must be that at least one other column of $B_{w-1} \ldots B_1 B_0$ is nonzero besides B_0; thus,

$$\mathsf{wt}(\mathcal{B}) \geq \mathsf{wt}\big(\mathcal{B}'\big) + 2.$$

Now, since $(\mathcal{A})_2 = (\mathcal{B})_2$, we have

$$(\mathcal{A}' \| \underbrace{A_{w-1} \ldots A_2 A_1}_{\text{zero columns}} A_0)_2 = (\mathcal{B}' \| B_{w-1} \ldots B_2 B_1 B_0)_2$$
$$\implies (\mathcal{A}')_2 = (\mathcal{B}')_2 + \frac{(B_{w-1} \ldots B_2 B_1 B_0)_2 - (A_0)_2}{2^w}.$$

The expression $((B_{w-1} \ldots B_2 B_1 B_0)_2 - (A_0)_2) / 2^w$ must evaluate to a vector of integers. Coordinate j of this vector is computed like so:

$$\left((b_{j\ w-1} \ldots b_{j2} b_{j1} b_{j0})_2 - (a_{j0})_2\right) / 2^w.$$

This value cannot be negative. To see this, suppose $(b_{j\ w-1} \ldots b_{j2} b_{j1} b_{j0})_2 < (a_{j0})_2$. Then $(b_{j\ w-1} \ldots b_{j2} b_{j1} b_{j0})_2$ must represent some integer in E_w. This integer and a_{j0} are not equal, however, they are congruent modulo 2^w. But, this contradicts the fact that no two distinct integers in E_w are congruent modulo 2^w. So, it must be that $(b_{j\ w-1} \ldots b_{j2} b_{j1} b_{j0})_2 \geq (a_{j0})_2$. Let $d = 2^w - 1$. Now,

$$\begin{aligned} 0 \leq (b_{j\ w-1} \ldots b_{j2} b_{j1} b_{j0})_2 - (a_{j0})_2 &\leq (\underbrace{d \ldots dd}_{w})_2 - a_{j0} \\ &< (\underbrace{d \ldots dd}_{w})_2 \\ &= (2^w - 1)d \\ &< 2^w d. \end{aligned}$$

Let $\widehat{B}_w = ((B_{w-1} \ldots B_2 B_1 B_0)_2 - (A_0)_2) / 2^w$. From the inequality above, we see that each coordinate of $\widehat{B}_w$ is a digit in E_w.

We have

$$(\mathcal{A}')_2 = (\mathcal{B}')_2 + \widehat{B}_w.$$

Using an argument similar to the proof of Lemma 4.10, we can perform the addition operation on the right-hand side of this equation and construct a representation $(\mathcal{B}'')_2 = (\mathcal{B}')_2 + \widehat{B}_w$ such that $\mathsf{wt}(\mathcal{B}'') \leq \mathsf{wt}(\mathcal{B}') + 1$.

Now, putting everything together we have

$$\begin{aligned} &\mathsf{wt}(\mathcal{B}) < \mathsf{wt}(\mathcal{A}) \\ \Longrightarrow\ &\mathsf{wt}\big(\mathcal{B}'\big) + 2 < \mathsf{wt}\big(\mathcal{A}'\big) + 1 \\ \Longrightarrow\ &\mathsf{wt}\big(\mathcal{B}'\big) + 1 < \mathsf{wt}\big(\mathcal{A}'\big) \\ \Longrightarrow\ &\mathsf{wt}\big(\mathcal{B}''\big) < \mathsf{wt}\big(\mathcal{A}'\big)\,. \end{aligned}$$

However, $(\mathcal{B}'')_2 = (\mathcal{A}')_2$ and $(\mathcal{A}')_2$ is a colexicographically smallest representation. Since $(\mathcal{A}')_2$ is shorter than $(\mathcal{A})_2$ this contradicts the minimality of ℓ. □

4.4.2 The Digits $\{0, 1, 3\}$

It is possible that there may be a simple strategy for building minimal weight $\{0, 1, 3\}$-joint representations, however, the only thing we can say with certainty about such a strategy is that it is not the one that builds a colexicographically smallest representation.

Example 4.16. Suppose $(\ldots A_2 A_1 A_0)_2$ is a colexicographically smallest representation of $N = (5, 9)^T$ which uses the digits $\{0, 1, 3\}$. If we were trying to construct this representation, we would first try to make A_0 a zero column. However, since both 5 and 9 are odd, this is not possible. So, we try to make A_1 a zero column. This can only be done by setting A_0 to $(1, 1)^T = (5, 9)^T \bmod 4$. If we continue in this manner, we arrive at the following representation:

$$\begin{pmatrix} 5 \\ 9 \end{pmatrix} = \begin{pmatrix} 0101 \\ 1001 \end{pmatrix}_2 .$$

This is the colexicographically smallest representation of $(5, 9)^T$ and it has weight 3. However,

$$\begin{pmatrix} 5 \\ 9 \end{pmatrix} = \begin{pmatrix} 0013 \\ 0033 \end{pmatrix}_2$$

and this representation has weight 2. So, for the digits $\{0, 1, 3\}$, the strategy of building a colexicographically smallest representation does not necessarily give a minimal weight representation. ◊

4.5 Signed Digits

For any integer $w \geq 2$, we define the set of digits

$$F_w := \{n \text{ mods } 2^w : n \in \mathbb{Z}\} = \{0, \pm 1, \pm 2, \ldots, \pm(2^{w-1} - 1)\} \cup \{2^{w-1}\}.$$

We use the convention that "n mods 2^w" returns a residue of n modulo 2^w of least absolute value, but when there are two such residues, it returns the positive one.

Example 4.17. For $w = 3$, we have

$$F_3 = \{0, \pm 1, \pm 2, \pm 3, 4\},$$

while

$$D_3 = \{0, \pm 1, \pm 3\}.$$

$\Diamond$

Many of the results we presented for the digits E_w in the previous section also hold for F_w.

Lemma 4.18. *Let $r \geq 1$ be an integer. If $(\mathcal{A})_2$ is a colexicographically smallest representation of a vector $N \in \mathbb{Z}^{r \times 1}$ which uses the digits F_w, then every nonzero column of $(\mathcal{A})_2$ must contain an odd digit.*

Proof. Let $\mathcal{A} = \ldots A_2 A_1 A_0$ and suppose $\mathcal{A}$ contains a nonzero column consisting of only even digits. By Lemma 4.7, we can assume that A_0 is such a nonzero column. This implies that N is composed of only even integers. Consider the $\{0, \pm 1\}$-joint representation of N formed from the $\{0, 1\}$- or $\{0, -1\}$-radix 2 representation of each coordinate of N (in fact, any $\{0, \pm 1\}$-joint representation of N will do). The least significant column of this representation must be a zero column. However, this contradicts the fact that $(\mathcal{A})_2$ is a colexicographically smallest representation of N. $\square$

Lemma 4.19. *Let $r \geq 1$ be an integer. Any vector $N \in \mathbb{Z}^{r \times 1}$ has a minimal weight F_w-joint representation where each nonzero column contains an odd digit.*

The proof of Lemma 4.19 is almost identical to that of Lemma 4.10 and so we omit it.

It will come as no surprise that we can build a colexicographically smallest representation of any vector $N \in \mathbb{Z}^{r\times 1}$ using the following algorithm:

Algorithm 4.20: COLEXI-SMALLEST-F_w-JOINT-REP(N)

```
comment: for any $n \in \mathbb{Z}$, $-2^{w-1} < n \text{ mods } 2^w \leq 2^{w-1}$.
$\mathcal{A} \leftarrow \epsilon$
while $N \neq \vec{0}$
  do { if $N \bmod 2 \neq \vec{0}$
         then $A_i \leftarrow N \text{ mods } 2^w$
         else $A_i \leftarrow \vec{0}$
       $\mathcal{A} \leftarrow A_i \,\|\, \mathcal{A}$
       $N \leftarrow (N - A_i)/2$
return $\mathcal{A}$
```

Example 4.21. We can visualize the steps of the algorithm by considering a window of width-w that slides over the $\{0,1\}$-joint representation of a vector of nonnegative integers. As the contents of the window are updated the $\{0,1\}$-joint representation can be affected by carry digits. When $N = (34, 55)^T$ and $w = 3$ we have

$$\begin{pmatrix}34\\55\end{pmatrix} = \begin{pmatrix}0100\boxed{010}\\0110\boxed{111}\end{pmatrix}_2 = \begin{pmatrix}0\boxed{100}002\\0\boxed{111}00\bar{1}\end{pmatrix}_2 = \begin{pmatrix}0004002\\100\bar{1}00\bar{1}\end{pmatrix}_2. \qquad \lozenge$$

Notice that if $\mathcal{A} = \ldots A_2 A_1 A_0$ is an output of this algorithm then any nonzero column is immediately followed by $w-1$ zero columns. From this property, we see that the output of the algorithm on input $N \in \mathbb{Z}^{r\times 1}$ is the *unique* colexicographically smallest representation of N.

Lemma 4.22. *Every $r \times 1$ vector N of nonnegative integers has a unique colexicographically smallest representation which uses the digits F_w.*

Proof. The zero vector has exactly one F_w-joint representation, so we can assume that N is not the zero vector. Let $(\mathcal{A})_2$ and $(\mathcal{B})_2$ be colexicographically smallest joint representations of N. We suppose $\mathcal{A} \neq \mathcal{B}$ and then show that this leads to a contradiction.

Let ℓ and ℓ' be the respective lengths of $\mathcal{A}$ and $\mathcal{B}$. Write

$$\mathcal{A} = A_{\ell-1} \dots A_2 A_1 A_0, \quad \text{and} \quad \mathcal{B} = B_{\ell'-1} \dots B_2 B_1 B_0.$$

We can assume that ℓ is as small as possible (i.e., there is no other choice of N which leads to a shorter $\mathcal{A}$).

Let

$$\mathcal{A}' = A_{\ell-1} \dots A_2 A_1, \quad \text{and} \quad \mathcal{B}' = B_{\ell'-1} \dots B_2 B_1.$$

If $A_0 = B_0$, then $(\mathcal{A}')_2 = (\mathcal{B}')_2$. So, by Lemma 4.7, $(N - A_0)/2$ has two different colexicographically smallest representations, contrary to the minimality of ℓ. Thus $A_0 \neq B_0$.

Neither of A_0 or B_0 can be a zero column. For, if B_0 is a zero column, then since $\mathcal{A} \preceq \mathcal{B}$, it must be that A_0 is a zero column. But this contradicts the fact that $A_0 \neq B_0$. So, B_0 is a nonzero column and the same argument shows that A_0 is a nonzero column.

By Lemma 4.18, one of the coordinates of N is an odd integer. Consider the output of Algorithm 4.20 on input N. The least significant column of the returned representation is nonzero and is followed by $w - 1$ zero columns. This representation cannot be colexicographically smaller than $\mathcal{A}$ or $\mathcal{B}$, so it must be that

$$\mathcal{A} = A_{\ell-1} \dots \underbrace{A_{w-1} \dots A_2 A_1}_{\text{zero columns}} A_0, \quad \text{and} \quad \mathcal{B} = B_{\ell'-1} \dots \underbrace{B_{w-1} \dots B_2 B_1}_{\text{zero columns}} B_0;$$

otherwise, $\mathcal{A}$ and $\mathcal{B}$ would not be colexicographically smallest representations. Now, from the value of $N \bmod 2^w$, we can conclude that $A_0 = B_0$; however, this contradicts the minimality of ℓ.

Thus, it must be that $\mathcal{A} = \mathcal{B}$; that is, there is exactly one colexicographically smallest representation of N. $\square$

4.5.1 Minimality of Colexi Smallest Representations

Theorem 4.23. *Let $r \geq 1$ be an integer. Then, for any $N \in \mathbb{Z}^{r \times 1}$, the colexicographically smallest F_w-joint representation of N has minimal weight.*

Proof. Let $(\mathcal{A})_2$ be the colexicographically smallest representation of N. Suppose $(\mathcal{A})_2$ does not have minimal weight. Then there is a minimal weight representation $(\mathcal{B})_2 = N$ with $\mathsf{wt}(\mathcal{B}) < \mathsf{wt}(\mathcal{A})$. By Lemma 4.19, we can assume that every nonzero column of $(\mathcal{B})_2$ contains an odd digit.

Let ℓ and ℓ' be the respective lengths of $\mathcal{A}$ and $\mathcal{B}$. Write

$$\mathcal{A} = A_{\ell-1} \ldots A_2 A_1 A_0, \quad \text{and} \quad \mathcal{B} = B_{\ell'-1} \ldots B_2 B_1 B_0.$$

We can assume that ℓ is as small as possible (i.e., there is no other choice of N which gives a shorter $\mathcal{A}$).

Since ℓ is minimal, it must be that $A_0 \neq B_0$. If B_0 is a zero column, then, because $\mathcal{A} \preceq \mathcal{B}$, A_0 must also be a zero column; however, this contradicts the fact that $A_0 \neq B_0$. So, B_0 is a nonzero column. Since B_0 contains an odd digit, some coordinate of N must be odd. This implies that A_0 is nonzero.

Let

$$\mathcal{A}' = A_{\ell-1} \ldots A_{w+1} A_w, \quad \text{and} \quad \mathcal{B}' = B_{\ell'-1} \ldots B_{w+1} B_w.$$

Because $(\mathcal{A})_2$ is the colexicographically smallest representation of N and A_0 is nonzero, all of $A_{w-1} \ldots A_2 A_1$ must be zero columns. So,

$$\mathsf{wt}(\mathcal{A}) = \mathsf{wt}\big(\mathcal{A}'\big) + 1.$$

Also, if B_0 is the only nonzero column of $B_{w-1} \ldots B_1 B_0$, then from the value of $N \bmod 2^w$ we could deduce that $A_0 = B_0$. However, since $A_0 \neq B_0$, it must be that at least one other column of $B_{w-1} \ldots B_1 B_0$ is nonzero besides B_0. Thus,

$$\mathsf{wt}(\mathcal{B}) \geq \mathsf{wt}\big(\mathcal{B}'\big) + 2.$$

Now, since $(\mathcal{A})_2 = (\mathcal{B})_2$ we have

$$(\mathcal{A}' \| \underbrace{A_{w-1} \dots A_2 A_1}_{\text{zero columns}} A_0)_2 = (\mathcal{B}' \| B_{w-1} \dots B_2 B_1 B_0)_2$$
$$\implies (\mathcal{A}')_2 = (\mathcal{B}')_2 + \frac{(B_{w-1} \dots B_2 B_1 B_0)_2 - (A_0)_2}{2^w}.$$

The expression $((B_{w-1} \dots B_2 B_1 B_0)_2 - (A_0)_2) / 2^w$ must evaluate to a vector of integers. Coordinate j of this vector is computed like so:

$$\left((b_{j\ w-1} \dots b_{j2} b_{j1} b_{j0})_2 - (a_{j0})_2\right) / 2^w.$$

Up to this point, our proof has essentially been the same as the proof of Theorem 4.15, however, now it differs slightly. Let $d = 2^{w-1}$. Now,

$$\begin{aligned} \left|(b_{j\ w-1} \dots b_{j2} b_{j1} b_{j0})_2 - (a_{j0})_2\right| &\le \left|(b_{j\ w-1} \dots b_{j2} b_{j1} b_{j0})_2\right| + \left|(a_{j0})_2\right| \\ &\le (\underbrace{d \dots dd}_{w})_2 + d \\ &= 2^w d. \end{aligned}$$

So we see that

$$-d \le \left((b_{j\ w-1} \dots b_{j2} b_{j1} b_{j0})_2 - (a_{j0})_2\right) / 2^w \le d.$$

If we examine the lower bound more closely, we see that the only possible way

$$-2^{w-1} = \left((b_{j\ w-1} \dots b_{j2} b_{j1} b_{j0})_2 - (a_{j0})_2\right) / 2^w$$

is if all of the b_{ji}'s are equal to -2^{w-1} and $a_{j0} = 2^{w-1}$. However, this cannot be the case since -2^{w-1} is not included in the set F_w. Thus,

$$-2^{w-1} < \left((b_{j\ w-1} \dots b_{j2} b_{j1} b_{j0})_2 - (a_{j0})_2\right) / 2^w \le 2^{w-1}.$$

Let $\widehat{B}_w = ((B_{w-1} \dots B_2 B_1 B_0)_2 - (A_0)_2) / 2^w$. From the bound above, we see that each coordinate of $\widehat{B}_w$ is a digit in F_w.

Now, our argument continues, to its conclusion, in the same way as the proof Theorem 4.15. Thus, we see that the colexicographically smallest representation does indeed have minimal weight. □

4.5.2 The Digits $\{0, \pm 1\}$

We now know that colexicographically smallest joint representations which use the digits F_w have minimal weight, however, we have not said anything about joint representation which use the digits $\{0, \pm 1\}$. In fact, the same property is true for these representations.

In our previous proofs of minimality, we relied on the fact that the digit sets we considered consisted of a complete system of residues modulo 2^w, for some $w \geq 2$. This is not true for the digits $\{0, \pm 1\}$ and so our previous arguments are not directly applicable. However, the statement above can be proven using results due to Grabner, Heuberger and Prodinger [12].

Grabner et al. give a very simple r-row analogue of the joint sparse form. To describe their representations, we need some notation. If $(\ldots A_2 A_1 A_0)_2$ is an r-row $\{0, \pm 1\}$-joint representation, then for a column A_i we define

$$A_i(\{\pm 1\}) := \{j \in \mathbb{Z} : a_{ji} \in \{\pm 1\}\}.$$

Thus, $A_i(\{\pm 1\})$ is the subset of $\{1, 2, \ldots, r\}$ which identifies the nonzero digits in column A_i. For example, if $A_0 = (1, \overline{1}, 0, 0, \overline{1})^T$, then $A_0(\{\pm 1\}) = \{1, 2, 5\}$. Note that if A_i is a zero column then $A_i(\{\pm 1\}) = \varnothing$.

Theorem 4.24 (Grabner, Heuberger, Prodinger). *Let $r \geq 1$ be an integer. Then any $N \in \mathbb{Z}^{r \times 1}$ has exactly one $\{0, \pm 1\}$-joint representation, $(\ldots A_2 A_1 A_0)_2$, with the property that, for any $i \geq 0$, either*

1. *$A_{i+1}(\{\pm 1\})$ is a proper superset of $A_i(\{\pm 1\})$, or*
2. *$A_{i+1}(\{\pm 1\}) = \varnothing$.*

Moreover, this representation has minimal weight.

If a $\{0, \pm 1\}$-joint representation satisfies the properties above we call it a GHP-joint form.

Example 4.25. Here is the GHP-joint form for $N = (55, 89, 144)^T$:

$$\begin{pmatrix} 55 \\ 89 \\ 144 \end{pmatrix} = \begin{pmatrix} 1\bar{1}0\bar{1}100\bar{1} \\ 1\bar{1}011001 \\ 10010000 \end{pmatrix}_2$$

Notice, for example, that for the leftmost two nonzero columns we have $A_6(\{\pm 1\}) = \{1, 2\}$ and $A_7(\{\pm 1\}) = \{1, 2, 3\}$. Thus $A_7(\{\pm 1\})$ is a proper superset of $A_6(\{\pm 1\})$, as required. ◊

If N is a vector of r nonnegative integers, then the GHP-joint form of N can be constructed by sliding an r-column window from right to left across the $\{0, 1\}$-joint representation of N. A GHP-joint form has the property that of any $r + 1$ consecutive columns, at most r are nonzero.

Corollary 4.26. *Let $r \geq 1$ be an integer. Then, for any $N \in \mathbb{Z}^{r \times 1}$, a colexicographically smallest $\{0, \pm 1\}$-joint representation of N has minimal weight.*

We call this result a "corollary" since it is established by making only minor changes to the argument Grabner et al. use to prove that the GHP-joint form has minimal weight.

Proof. Let $(\ldots A_2 A_1 A_0)_2$ be a colexicographically smallest representation of N. We claim that, for any $i \geq 0$, we may assume

$$A_{i+1}(\{\pm 1\}) \text{ is a superset of } A_i(\{\pm 1\}), \text{ or } A_{i+1}(\{\pm 1\}) = \varnothing.$$

To see this, observe that if $(\ldots A_2 A_1 A_0)_2$ has two consecutive nonzero columns where a row of these two columns equals 01 or $0\bar{1}$, these digits can be replaced with $1\bar{1}$ or $\bar{1}1$, respectively. Notice that such digit replacements have no affect on the colexicographic rank of $(\ldots A_2 A_1 A_0)_2$. We show that $(\ldots A_2 A_1 A_0)_2$ must be the GHP-joint form of N and then minimality follows.

Let i be the smallest integer such that $A_{i+1}(\{\pm 1\}) = A_i(\{\pm 1\}) \neq \varnothing$; that is, $A_{i+1}(\{\pm 1\})$ is a superset of $A_i(\{\pm 1\})$, but not a *proper* superset. If

i does not exist then $(\ldots A_2 A_1 A_0)_2$ is the GHP-joint form of N and we are done. Let A_t be the first zero column to the left of A_i.

Let

$$A_{i+1} = \begin{pmatrix} a_{1\ i+1} \\ a_{2\ i+1} \\ \vdots \\ a_{r\ i+1} \end{pmatrix}, \quad \text{and} \quad A_i = \begin{pmatrix} a_{1i} \\ a_{2i} \\ \vdots \\ a_{ri} \end{pmatrix}.$$

Column A_i is nonzero and so it must contain a nonzero digit. By possibly permuting the rows of $(\ldots A_2 A_1 A_0)_2$, we may assume that $a_{1i} \neq 0$. Since a_{1i} is nonzero and $A_{i+1}(\{\pm 1\}) = A_i(\{\pm 1\})$, $a_{1\ i+1}$ must also be nonzero.

Suppose $a_{1i} = 1$. In row 1, consider the maximal length run of ones that begins with $a_{1i} = 1$ and extends left. This run of ones is terminated by the digit 0 or -1. Because A_t is a zero column, this run of ones terminates before column t. The strings

$$01\ldots11, \text{ and } \bar{1}1\ldots11$$

can be replaced with

$$10\ldots0\bar{1}, \text{ and } 00\ldots0\bar{1},$$

respectively. An analogous string replacement can be done if a_{1i} was instead equal to -1. For each nonzero digit in column i, we apply this string replacement. We denote the resulting representation by $(\ldots A'_2 A'_1 A'_0)_2$.

Because of the string replacements, A'_{i+1} is a zero column, whereas A_{i+1} is a nonzero column. However, this contradicts the fact that $(\ldots A_2 A_1 A_0)_2$ is a colexicographically smallest representation. Thus, there can be no i for which $A_{i+1}(\{\pm 1\}) = A_i(\{\pm 1\}) \neq \varnothing$. So, $(\ldots A_2 A_1 A_0)_2$ is the GHP-joint form of N and therefore has minimal weight. □

From the previous argument we see that the GHP-joint form of N is a colexicographically smallest representation of N. A vector N can have many different colexicographically smallest representations, however, it has exactly one GHP-joint form.

4.5.3 The Digits $\{0, \pm 1, \pm 2, \ldots, \pm m\}$

Following Möller [28, 27], for any integer $m \geq 2$, we define

$$G_m := \{n \in \mathbb{Z} : |n| \leq m\} = \{0, \pm 1, \pm 2, \ldots, \pm m\}.$$

Note that m is not restricted to be a power of 2. We make the following conjecture about G_m-joint representations:

Conjecture 4.27. Let $r \geq 1$ be an integer. Then, for any $N \in \mathbb{Z}^{r\times 1}$, a colexicographically smallest G_m-joint representation of N has minimal weight.

When $r = 1$, a colexicographically smallest G_m-representation of an integer will not make use of any even digits (as in Lemmas 4.9 and 4.18). If we remove all the even nonzero digits from G_m we are left with what Möller calls the *signed fractional window digits*. If we let $w = \lfloor \lg m \rfloor + 1$, then Möller explains how to build G_m-representations by sliding a variable width window (having width either w or $w + 1$) across the $\{0, 1\}$-representation of a nonnegative integer. It is not difficult to prove that the representations Möller builds are colexicographically smallest. That these representations have minimal weight is proven in [27].

For general $r \geq 1$, the fact that G_m does not consist of a complete system of residues modulo 2^w means that the arguments we previously used to prove that colexicographically smallest representations have minimal weight are not readily applicable, and so we leave this as a conjecture for now.

An interesting special case is obtained when, for any $w \geq 2$, we take $m = 2^{w-1}$:

$$G_{2^{w-1}} = \{0, \pm 1, \pm 2, \ldots, \pm 2^{w-1}\}.$$

There is a very simple algorithm for constructing $G_{2^{w-1}}$-joint representations. It is based on a special signed binary representation that has been proposed by a number of different authors [13, 16, 19, 20, 36].

Suppose $n = (b_{\ell-1} \ldots b_1 b_0)_2$ with $b_i \in \{0, 1\}$. Consider the representation $(b'_\ell \ldots b'_1 b'_0)_2$ with $b'_i \in \{0, \pm 1\}$ which results from the *digit-wise* subtraction of $n = (b_{\ell-1} \ldots b_1 b_0)_2$ from $2n = (b_{\ell-1} \ldots b_1 b_0 0)_2$:

$$\begin{array}{cccccccc} & b_{\ell-1} & b_{\ell-2} & b_{\ell-3} & \ldots & b_0 & 0 \\ - & 0 & b_{\ell-1} & b_{\ell-2} & \ldots & b_1 & b_0 \\ \hline & b'_\ell & b'_{\ell-1} & b'_{\ell-2} & \ldots & b'_1 & b'_0 \end{array}$$

If we set $b_\ell = b_{-1} = 0$, then we have

$$b'_i = b_{i-1} - b_i, \text{ for } 0 \le i \le \ell.$$

Note that $(b'_\ell \ldots b'_1 b'_0)_2$ is a representation of n. The signs of the nonzero digits of $(b'_\ell \ldots b'_1 b'_0)_2$ must alternate as we read them from left to right or right to left. To see this, suppose $b'_v 00 \ldots 0 b'_u$ is a substring of $b'_\ell \ldots b'_1 b'_0$. Then,

$$b'_v + b'_u = \sum_{u \le i \le v} b'_i = \sum_{u \le i \le v} (b_{i-1} - b_i) = b_{u-1} - b_v \in \{0, \pm 1\}.$$

If $b'_u = b'_v = 1$ or $b'_u = b'_v = -1$, then this would contradict the equation above. So, if b_u and b_v are both nonzero, they must have opposite signs.

Consider any w consecutive digits (i.e., any window of width-w) taken from the representation $(b'_\ell \ldots b'_1 b'_0)_2$. If $b'_{i+w-1} \cdots b'_{i+1} b'_i$ is a substring of $b'_\ell \ldots b'_1 b'_0$, we would like to know what values $(b'_{i+w-1} \cdots b'_{i+1} b'_i)_2$ can take. The largest possible value is

$$2^{w-1} = (1\underbrace{0 \ldots 0}_{w-1})_2;$$

and the smallest possible value is

$$-2^{w-1} = (\bar{1}\underbrace{0 \ldots 0}_{w-1})_2.$$

All integers d with $0 \le d < 2^{w-1}$ are possible since their $\{0, 1\}$-representations have length at most $w - 1$ (and so their "sign-alternating" representations, as defined above, have length at most w). Clearly, the negative of these values are also possible. So, the set of possible values is exactly $G_{2^{w-1}}$.

For an $r \times 1$ vector N of nonnegative integers, let $(B_{\ell-1} \ldots B_1 B_0)_2$ be the $\{0,1\}$-joint representation of N. We take B_ℓ and B_{-1} to be zero columns and then define

$$B'_i = B_{i-1} - B_i, \text{ for } 0 \leq i \leq \ell.$$

Thus, each row of $(B'_\ell \ldots B'_1 B'_0)_2$ is the sign-alternating representation of a coordinate of N. Now, we build a $G_{2^{w-1}}$-joint representation of N by sliding a w-column window from right to left across the representation $(B'_\ell \ldots B'_1 B'_0)_2$.

Example 4.28. Suppose $w = 4$. Then, $G_{2^3} = \{0, \pm1, \pm2, \ldots, \pm8\}$. Below, we list the $\{0,1\}$-joint representation of $N = (233, 377, 987)^T$ and its sign-alternating joint representation. We show the digit substitutions made as the window slides right to left.

$$\begin{aligned}
\begin{pmatrix} 233 \\ 377 \\ 987 \end{pmatrix} = \begin{pmatrix} 0011101001 \\ 0101111001 \\ 1111011011 \end{pmatrix}_2 &= \begin{pmatrix} 00010\bar{1}1\boxed{\bar{1}01\bar{1}} \\ 001\bar{1}1000\boxed{\bar{1}01\bar{1}} \\ 0100\bar{1}10\boxed{\bar{1}10\bar{1}} \end{pmatrix}_2 \\
&= \begin{pmatrix} 0001\boxed{00\bar{1}1}000\bar{7} \\ 001\bar{1}\boxed{1000}000\bar{7} \\ 0100\boxed{0\bar{1}10}000\bar{5} \end{pmatrix}_2 \\
&= \begin{pmatrix} \boxed{0001}000\bar{1}000\bar{7} \\ \boxed{001\bar{1}}0008000\bar{7} \\ \boxed{0100}000\bar{2}000\bar{5} \end{pmatrix}_2 \\
&= \begin{pmatrix} 0001000\bar{1}000\bar{7} \\ 00010008000\bar{7} \\ 0004000\bar{2}000\bar{5} \end{pmatrix}_2 .
\end{aligned}$$

$\Diamond$

It seems likely that any $G_{2^{w-1}}$-joint representation constructed in this manner will be a colexicographically smallest representation of N, however, this remains to be proven.

The window which slides over the representation $(B'_\ell \ldots B'_1 B'_0)_2$ can also travel *left to right*. This will construct a joint representation from the opposite direction. If we can prove that sliding the window right to left produces a minimal weight representation then is it a simple matter to prove that the left to right version also produces a minimal weight representation using an argument by Avanzi [1]. [2]

For general $m \geq 2$, it is also possible to use the representation $(B'_\ell \dots B'_1 B'_0)_2$ to build G_m-joint representations; we just allow the width of the window to change as it slides across the representation.

[2]A nice application of Avanzi's "input reversing argument" can be found in [27].

Chapter 5

Nonadjacent Digit Sets

5.1 Introduction and History

In 1960, through his investigations on how to reduce the number of additions and subtractions used in binary multiplication and division, Reitwiesner [39] gave a constructive proof that every integer has a unique $\{0, 1, -1\}$-radix 2 representation with a *minimal* number of nonzero digits.

Reitwiesner's canonical representations have a simple description. A $\{0, 1, -1\}$ radix 2 representation of an integer is in Reitwiesner's canonical form if and only if it satisfies the following property:

NA-1 *Of any two consecutive digits, at most one is nonzero.*

Said another way, for such representations, nonzero digits are nonadjacent. These representations have come to be called *nonadjacent forms* (NAFs).

If a finite length radix 2 representation has digit set D and satisfies **NA-1**, we call it a *D-nonadjacent form* (D-NAF). In this chapter, we consider the question of which sets D provide nonadjacent forms for *every positive* integer. If D is such a digit set then we call it a *nonadjacent digit set* (NADS).

A related question has been studied by Matula. In [24], Matula defines and investigates *basic* digit sets. A set of digits containing 0 is called *basic* if it provides every integer, positive and negative, with a *unique* radix-r representation which does not include a seperate $+/-$ sign (i.e., for such digit sets, we can write down a representation of a negative number without having to put a minus sign in front of it). Matula shows that if a digit set is basic, then $r \neq 2$; in this chapter we are concerned only with radix 2 representations. Another difference between our work and Matula's is that he imposes no relation on the digits of a representation while we are interested only in nonadjacent representations.

We examine digit sets of the form $\{0, 1, x\}$ with $x \in \mathbb{Z}$. It is known that letting $x = -1$ gives a NADS, but it is somewhat surprising that there are many values of x with this property; for example, $x = -5, -13, -1145$. We give infinite families of x's for which $\{0, 1, x\}$ is a NADS, and we also give infinite families of x's for which $\{0, 1, x\}$ is not a NADS. We also give some results on the necessary conditions D must satisfy in order to be a NADS. The algorithms we present and analyze for computing D-NAFs might be of some interest as well.

5.2 Preliminaries

We start by introducing some definitions and notation which will facilitate our discussions.

We apply some of our terminology for representations to strings. If $0 \in D$ and a finite string $\alpha \in D^*$ satisfies the property **NA-1**, then we call α a D-NAF. If in addition, $(\alpha)_2 = n$ we say α is a D-NAF for n. Notice that if α is a D-NAF for n then α with any leading zeros removed is also a D-NAF for n. We denote the string formed by deleting the leading zeros from α by $\widehat{\alpha}$.

Given a digit set D and an integer n, we define a map

$$R_D(n) := \begin{cases} \widehat{\alpha} & \text{where } \alpha \in D^* \text{ is a } D\text{-NAF for } n\text{, if one exists} \\ \perp & \text{otherwise.} \end{cases}$$

Here, $\perp$ is just some symbol not in D. If $R_D(n)$ evaluates to a D-NAF for n, then by definition that string has no leading zeros. For example, if $D = \{0, 1, -9\}$ then $R_D(7)$ might evaluate to $1000\overline{9}$ since $1000\overline{9}$ is a D-NAF, has no leading zeros, and $(1000\overline{9})_2 = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 - 9 \cdot 2^0 = 7$. If there is more than one string in D which is a D-NAF for n and has no leading zeros then $R_D(n)$ might evaluate to any one of these strings. Later on we will prove that 3 does not have a D-NAF, hence $R_D(3) = \perp$.

We are interested in determining which integers have D-NAFs, so we define the set

$$\mathrm{NAF}(D) := \{n \in \mathbb{Z} : R_D(n) \neq \perp\} .$$

From our example with $D = \{0, 1, 9\}$ we see $7 \in \mathrm{NAF}(D)$ but $3 \notin \mathrm{NAF}(D)$. Using this notation, our definition of a nonadjacent digit set is as follows:

Definition 5.1. D is a *nonadjacent digit set* if $\mathbb{Z}^+ \subseteq \mathrm{NAF}(D)$.

5.3 Necessary Conditions for $\{0, 1, x\}$ to be a NADS

If we suppose $D = \{0, 1, x\}$ is a nonadjacent digit set then we can deduce necessary conditions on x.

Theorem 5.2. *Let $D = \{0, 1, x\}$. If there exists $n \in \mathrm{NAF}(D)$ with $n \equiv 3 \pmod 4$, then $x \equiv 3 \pmod 4$.*

Proof. Take $n \in \mathrm{NAF}(D)$ with $n \equiv 3 \pmod 4$. For some particular D-NAF, say $(\ldots a_2 a_1 a_0)_2$, we have

$$\begin{aligned}(\ldots a_2 a_1 a_0)_2 &= n \\ &\implies a_0 \equiv 1 \pmod 2 \\ &\implies a_0 \neq 0 .\end{aligned}$$

Since a_0 is nonzero and the representation is nonadjacent we have $a_1 = 0$. Thus

$$\begin{aligned}(\ldots a_2 0 a_0)_2 &= n \\ &\implies a_0 \equiv 3 \pmod 4 \\ &\implies a_0 \neq 1 \\ &\implies a_0 = x .\end{aligned}$$

So $x = a_0 \equiv 3 \pmod 4$. □

If $D = \{0,1,x\}$ is a NADS then $3 \in \mathrm{NAF}(D)$, and by the previous result $x \equiv 3 \pmod 4$. So, if we are trying to find a value of x that makes $\{0,1,x\}$ a NADS we need only consider those values congruent to 3 modulo 4.

5.3.1 The case $x > 0$

If we restrict x to be a positive integer, then we can give a complete characterization of all values which make $D = \{0,1,x\}$ a NADS. It is well known that $x = 3$ is such a value, and this is remarked by Solinas [42]. We give a proof of this fact and then show that no other positive value of x makes $\{0,1,x\}$ a NADS.

Theorem 5.3. *The only NADS of the form $\{0,1,x\}$ with $x > 0$ is $\{0,1,3\}$.*

Proof. Let n be any positive integer. We want to show that n has a $\{0,1,3\}$-NAF. Let $(\ldots a_2a_1a_0)_2$ be the usual $\{0,1\}$-radix 2 representation of n. If this representation satisfies **NA-1** there is nothing to prove, so suppose it does not. Let i be the smallest integer for which $a_{i+1} = a_i = 1$. Replace digits a_{i+1} and a_i by 0 and 3, respectively. Since $2^{i+1} + 2^i = 0 \cdot 2^{i+1} + 3 \cdot 2^i$, the resulting representation stands for the same integer. By working from right to left, repeating this substitution as necessary, we transform $(\ldots a_2a_1a_0)_2$ into a $\{0,1,3\}$-NAF. This proves that $\{0,1,3\}$ is a NADS.

Now consider x with $x > 3$. We show $n = 3$ does not have a $\{0,1,x\}$-NAF. Suppose to the contrary that for some $\{0,1,x\}$-NAF we have $(\ldots a_2a_1a_0)_2 = 3$. Since 3 is odd, $a_0 \neq 0$ and so $a_1 = 0$. Now $a_0 \equiv 3 \pmod 4$ so it must be that $a_0 = x$. However, since each of the digits in $\{0,1,x\}$ is nonnegative we have

$$3 = (\ldots a_2 0 x)_2 = \cdots + a_2 2^2 + 0 \cdot 2^1 + x \geq x > 3\,,$$

which is a contradiction. So, 3 does not have a $\{0,1,x\}$-NAF when $x > 3$. □

An example helps illustrate the construction used in the above proof. Suppose $n = 237$. To find a $\{0, 1, 3\}$-NAF for 237 we start with its usual binary representation and then, working from right to left, replace any occurrences of the digits 11 with 03:

$$237 = (11101101)_2 = (10300301)_2 \,.$$

A natural question to ask is if this is the only $\{0, 1, 3\}$-NAF for 237. We give the answer in the next section.

5.3.2 Uniqueness

We show that every integer, not only just the positive ones, has at most one $\{0, 1, x\}$-NAF where $x \equiv 3 \pmod 4$.

Theorem 5.4. *If $x \equiv 3 \pmod 4$, then any integer has at most one finite length $\{0, 1, x\}$-nonadjacent form.*

Proof. Let $D = \{0, 1, x\}$ and suppose the result is false. Then it must be that

$$(a_{\ell-1} \ldots a_2 a_1 a_0)_2 = (b_{\ell'-1} \ldots b_2 b_1 b_0)_2$$

where $(a_{\ell-1} \ldots a_2 a_1 a_0)_2$ and $(b_{\ell'-1} \ldots b_2 b_1 b_0)_2$ are two different D-NAFs with lengths ℓ and ℓ' respectively. These representations stand for the same integer, call it n. We can assume that ℓ is as small as possible.

If $a_0 = b_0$, then

$$(a_{\ell-1} \ldots a_2 a_1)_2 = (b_{\ell'-1} \ldots b_2 b_1)_2 \,,$$

and so we have two different, and shorter, D-NAFs which stand for the same integer, contrary to the minimality of ℓ. So it must be that $a_0 \neq b_0$.

If one of a_0 or b_0 is 0, then n is even, and so both a_0 and b_0 are 0. But a_0 and b_0 are different so it must be that a_0 is equal to 1 or x. Without loss of generality, we can assume the representations have the form

$$(a_{\ell-1} \ldots a_2 0 x)_2 = (b_{\ell'-1} \ldots b_2 0 1)_2 \,.$$

This implies $x \equiv 1 \pmod 4$, contrary to our hypothesis that $x \equiv 3 \pmod 4$. Thus every integer has at most one D-NAF. $\square$

5.4 Recognizing NADS of the form $\{0, 1, x\}$

From now on we fix $D = \{0, 1, x\}$ with $x \equiv 3 \pmod 4$. In this section we work towards a method of deciding if $\{0, 1, x\}$ is a NADS. By Theorem 5.3, this is easy when $x > 0$, so we will assume $x < 0$.

Recall that $R_D(n)$ either evaluates to the symbol $\perp$ or a finite string, with no leading zeros, that is a D-NAF for n. Theorem 5.4 tells us that any n has at most one D-NAF, so in the second case, the string returned by $R_D(n)$ is unique. Thus, $R_D(n)$ is well defined (i.e., for every input n there is exactly one output.).

The ability to evaluate $R_D(n)$ can be useful in deciding if D is a NADS. If we can find $n \in \mathbb{Z}^+$ such that $R_D(n) = \perp$ then we know that D is not a NADS. Also, if we have an algorithmic description of $R_D(n)$, we might be able to analyze this algorithm and show that for any $n \in \mathbb{Z}^+$, $R_D(n) \neq \perp$, thus proving that D is a NADS.

We show that $R_D(n)$ can be computed recursively and give an algorithm which evaluates $R_D(n)$ in this manner. We begin with some lemmas:

Lemma 5.5. *If $n \equiv 0 \pmod 4$ then $n \in \mathrm{NAF}(D)$ if and only if $n/4 \in \mathrm{NAF}(D)$. Further, if $n \in \mathrm{NAF}(D)$ then $R_D(n) = R_D(n/4)\|00$.*

Proof. Since $n \equiv 0 \pmod 4$, the definition of the digit set D implies that any D-NAF for n is of the form $(a_{\ell-1} \ldots a_3 a_2 00)_2$, where $a_{\ell-1} \neq 0$. Now,

$$\begin{aligned} n \in \mathrm{NAF}(D) &\iff n \text{ has a } D\text{-NAF of the form } (a_{\ell-1} \ldots a_3 a_2 00)_2 \\ &\iff n/4 \text{ has a } D\text{-NAF of the form } (a_{\ell-1} \ldots a_3 a_2)_2 \\ &\iff n/4 \in \mathrm{NAF}(D)\,, \end{aligned}$$

which proves the first part of the lemma. If $n \in \mathrm{NAF}(D)$ then

$$R_D(n) = a_{\ell-1} \ldots a_3 a_2 00 = a_{\ell-1} \ldots a_3 a_2 \| 00 = R_D(n/4)\|00\,,$$

which proves the second part of the lemma. □

We omit the proofs of the next three lemmas since they can be established by making only minor changes to the proof of Lemma 5.5.

Lemma 5.6. *If $n \equiv 1 \pmod 4$ then $n \in \mathrm{NAF}(D)$ if and only if $(n-1)/4 \in \mathrm{NAF}(D)$. Further, if $n \in \mathrm{NAF}(D)$ then $R_D(n) = R_D(\frac{n-1}{4})\|01$.*

Lemma 5.7. *If $n \equiv 2 \pmod 4$ then $n \in \mathrm{NAF}(D)$ if and only if $n/2 \in \mathrm{NAF}(D)$. Further, if $n \in \mathrm{NAF}(D)$ then $R_D(n) = R_D(n/2)\|0$.*

Lemma 5.8. *If $n \equiv 3 \pmod 4$ then $n \in \mathrm{NAF}(D)$ if and only if $(n-x)/4 \in \mathrm{NAF}(D)$. Further, if $n \in \mathrm{NAF}(D)$ then $R_D(n) = R_D(\frac{n-x}{4})\|0x$.*

Given an integer n, if we somehow know that $n \in \mathrm{NAF}(D)$ then Lemmas 5.5–5.8 suggest a recursive procedure that we can use to evaluate $R_D(n)$. To illustrate suppose $D = \{0, 1, -9\}$. It was shown in an earlier example that $7 \in \mathrm{NAF}(D)$. Using these lemmas, we have:

$$R_D(7) = R_D(4)\|0\overline{9} = R_D(1)\|00\|0\overline{9} = 1\|00\|0\overline{9} = 1000\overline{9}\,.$$

To describe the general procedure for computing $R_D(n)$, given that $n \in \mathrm{NAF}(D)$, we use the following two functions:

$$f_D(n) := \begin{cases} n/4 & \text{if } n \equiv 0 \pmod 4 \\ (n-1)/4 & \text{if } n \equiv 1 \pmod 4 \\ n/2 & \text{if } n \equiv 2 \pmod 4 \\ (n-x)/4 & \text{if } n \equiv 3 \pmod 4\,, \end{cases} \tag{5.1}$$

$$g_D(n) := \begin{cases} 00 & \text{if } n \equiv 0 \pmod 4 \\ 01 & \text{if } n \equiv 1 \pmod 4 \\ 0 & \text{if } n \equiv 2 \pmod 4 \\ 0x & \text{if } n \equiv 3 \pmod 4\,. \end{cases} \tag{5.2}$$

Note that f_D returns an integer, and g_D returns a string. Here is the procedure described in pseudocode:

Procedure 5.9: $\text{EVAL}_\alpha\text{-}R_D(n)$

```
$\alpha \leftarrow \epsilon$
**while** $n \neq 0$
    **do** $\begin{cases} \alpha \leftarrow g_D(n) \parallel \alpha \\ n \leftarrow f_D(n) \end{cases}$
**return** $\widehat{\alpha}$
```

Procedure 5.9 terminates on input n if and only if ${f_D}^i(n) = 0$ for some positive integer i. An easy calculation shows that, for $D = \{0, 1, -9\}$, ${f_D}^3(7) = 0$, and so the procedure terminates on input $n = 7$. However, $f_D(3) = 3$ and so ${f_D}^i(3) = 3 \neq 0$ for all i, thus the procedure does not terminate on input $n = 3$.

Using the previous lemmas, we can show Procedure 5.9 terminates on input n if and only if $n \in \text{NAF}(D)$. Instead of making use of the lemmas individually, it is more convenient to summarize them as follows:

Lemma 5.10. *For all $n \in \mathbb{Z}$, $n \in \text{NAF}(D)$ if and only if $f_D(n) \in \text{NAF}(D)$. Further, if $n \in \text{NAF}(D)$ then $R_D(n) = R_D(f_D(n)) \| g_D(n)$.*

Now, suppose $n \in \text{NAF}(D)$. Then the finite string $R_D(n)$ can be computed with a finite number of recursive steps. This implies that there is some positive integer i such that ${f_D}^i(n) = 0$, which in turn implies that the procedure terminates. Conversely, suppose the procedure terminates. Then ${f_D}^i(n) = 0$ for some i, and clearly $0 \in \text{NAF}(D)$. Thus, ${f_D}^i(n) \in \text{NAF}(D)$, and by the lemma $n \in \text{NAF}(D)$.

Procedure 5.9 is named $\text{EVAL}_\alpha\text{-}R_D(n)$. We justify this name by noting that if the procedure terminates, it returns a string with no leading zeros (i.e., $\widehat{\alpha}$) equal to $R_D(n)$. We are not able to evaluate $R_D(n)$ for all values of n using this procedure because we have not yet described a way to recognize when $R_D(n) = \perp$. We proceed to do this now.

To decide if $D = \{0, 1, x\}$ is a NADS, it suffices to determine if there are any $n \in \mathbb{Z}^+$ for which Procedure 5.9 fails to terminate. We can determine if the procedure will terminate by examining the iterates of f_D.

Let n be a positive integer. Observe that, for $n \not\equiv 3 \pmod 4$, we have that

$$n > f_D(n) \geq 0\,, \tag{5.3}$$

and, for $n \equiv 3 \pmod 4$, that

$$n > f_D(n) \iff n > \frac{-x}{3} \tag{5.4}$$

$$f_D(n) \geq 0 \iff n \geq x\,. \tag{5.5}$$

Since x is negative, we see that any iterate of the function f_D, on input n, always results in a nonnegative integer. Consider the graph G_n having directed edges

$$n \to f_D(n) \to {f_D}^2(n) \to {f_D}^3(n) \to \cdots\ .$$

The vertices of G_n are nonnegative integers. Inequalities (5.3) and (5.4) tell us that there must be some vertex of G_n that is less than $\frac{-x}{3}$. Suppose ${f_D}^i(n) < \frac{-x}{3}$. We claim ${f_D}^{i+1}(n) < \frac{-x}{3}$ as well. This is clearly true if ${f_D}^i(n) \equiv 0,1,2 \pmod 4$. If ${f_D}^i(n) \equiv 3 \pmod 4$ then

$$\begin{aligned} {f_D}^i(n) < \frac{-x}{3} &\implies \frac{{f_D}^i(n) - x}{4} < \frac{\frac{-x}{3} - x}{4} \\ &\implies {f_D}^{i+1}(n) < \frac{-x - 3x}{12} = \frac{-x}{3}\,, \end{aligned}$$

and so the claim is true. The claim also tells us that if ${f_D}^i(n) < \frac{-x}{3}$, then any subsequent iterate of f_D must be less than $\frac{-x}{3}$.

From the preceding discussion it is clear that for a positive integer n, either:

1. G_n is a path terminating at 0, or
2. G_n contains a directed cycle of integers in the interval $\{1,2,\ldots,\lfloor\frac{-x}{3}\rfloor\}$.

If we can detect a directed cycle in G_n then we can determine whether or not Procedure 5.9 will terminate on input n. To do this we need to compute and store some of the vertices of G_n. However, as Procedure 5.9 executes, it computes all the vertices of G_n, so we might as well modify the procedure to detect a directed cycle in G_n on its own. This modification is described as Algorithm 5.11.

Algorithm 5.11: EVAL-$R_D(n)$

```
$\alpha \leftarrow \epsilon$
while $n > \frac{-x}{3}$
  do $\alpha \leftarrow g_D(n) \parallel \alpha$
     $n \leftarrow f_D(n)$
$\mathcal{S} \leftarrow \varnothing$
while $n \neq 0$
  do if $n \in \mathcal{S}$
       then return $\perp$
     $\mathcal{S} \leftarrow \mathcal{S} \cup \{n\}$
     $\alpha \leftarrow g_D(n) \parallel \alpha$
     $n \leftarrow f_D(n)$
return $\widehat{\alpha}$
```

Now we can use the title “Algorithm” rather than “Procedure”, because EVAL-$R_D(n)$ terminates for every $n \in \mathbb{Z}^+$. (For some positive integers, it was shown that EVAL$_\alpha$-$R_D(n)$ fails to terminate, which is why it cannot technically be called an algorithm.) As its name suggests, Algorithm 5.11 evaluates $R_D(n)$ for any $n \in \mathbb{Z}^+$. It is possible to show that the running time of EVAL-$R_D(n)$ is $O(\lg n + |x|)$.

Returning to our main task of recognizing when $\{0, 1, x\}$ is a NADS, Algorithm 5.11 and the preceding analysis are very helpful since they lead us to the following result:

Theorem 5.12. *Suppose x is a negative integer and $x \equiv 3 \pmod 4$. If every element in the set $\{n \in \mathbb{Z}^+ : n \leq \lfloor -x/3 \rfloor\}$ has a $\{0, 1, x\}$-NAF, then $\{0, 1, x\}$ is a NADS.*

Proof. From inspection of Algorithm 5.11 this result is almost immediate, however we can give a formal argument using the graph G_n.

Suppose the hypothesis is true. We must argue that $\{0, 1, x\}$ is a NADS. Take any $n \in \mathbb{Z}^+$ and consider the graph G_n. Suppose G_n contains a directed cycle. Let n_0 be a vertex in this cycle. Then $1 \leq n_0 \leq \lfloor -x/3 \rfloor$, and G_{n_0} must contain the same directed cycle. This implies that n_0 does not have a $\{0, 1, x\}$-NAF, contrary to our hypothesis. So, G_n is a path terminating at 0, and thus n has a $\{0, 1, x\}$-NAF. □

Theorem 5.12 suggests a computational method of determining if $\{0, 1, x\}$ is a NADS. For each $n \in \mathbb{Z}^+, n \leq \lfloor -x/3 \rfloor$, compute EVAL-$R_D(n)$. If all of these values have $\{0, 1, x\}$-NAFs then $\{0, 1, x\}$ is a NADS; otherwise, we find a value which does not have a $\{0, 1, x\}$-NAF which proves that $\{0, 1, x\}$ is not a NADS. To recognize a NADS, this method requires $\lfloor -x/3 \rfloor$ calls to EVAL-$R_D(n)$. However, we can decrease this number, as the next result shows.

Corollary 5.13. *Suppose x is a negative integer and $x \equiv 3 \pmod 4$. If every element in the set $\{n \in \mathbb{Z}^+ : n \leq \lfloor -x/3 \rfloor, n \equiv 3 \pmod 4\}$ has a $\{0, 1, x\}$-NAF, then $\{0, 1, x\}$ is a NADS.*

Proof. If $\{0, 1, x\}$ is not a NADS then choose the smallest integer $n_0 \in \mathbb{Z}^+$ such that G_{n_0} contains a directed cycle. By Theorem 5.12 it must be that $n_0 \leq \lfloor -x/3 \rfloor$. Let $n_1 = f_D(n_0)$, then (n_0, n_1) is an arc of G_n. If $n_0 \not\equiv 3 \pmod 4$ then $n_1 < n_0$ and G_{n_1} contains the same directed cycle, contrary to the choice of n_0. Thus, it must be that $n_0 \equiv 3 \pmod 4$. So, if the hypothesis is true, there can be no smallest positive integer which does not have a $\{0, 1, x\}$-NAF. Hence $\{0, 1, x\}$ is a NADS. □

Now we can detect a NADS of the form $\{0, 1, x\}$ with about $\lfloor -x/12 \rfloor$ calls to EVAL-$R_D(n)$. An optimized version of an algorithm which utilizes this method is described in Algorithm 5.14. We have used this algorithm to find all the values of x greater than -10^6 such that $\{0, 1, x\}$ is a NADS; some of these values are listed in Appendix B.

Algorithm 5.14: IS-NADS(x)

```
$N \leftarrow 3$
$\mathcal{T} \leftarrow \varnothing$
while $N \leq \frac{-x}{3}$
    do
        $n \leftarrow N$
        $\mathcal{S} \leftarrow \varnothing$
        while $n \neq 0$ and $n \notin \mathcal{T}$
            do
                if $n \in \mathcal{S}$
                    then return "no"
                $\mathcal{S} \leftarrow \mathcal{S} \cup \{n\}$
                $n \leftarrow f_D(n)$
        $N \leftarrow N + 4$
        $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{S}$
return "yes"
```

5.5 Directed Graphs and NADS

For small values of x, a convenient way to demonstrate that $\{0, 1, x\}$ is a NADS is to draw a number of directed graphs. From the previous section, we know that $\{0, 1, x\}$ is a NADS if and only if each directed graph, G_n, $n \in \{1, 2, \ldots, \lfloor \frac{-x}{3} \rfloor\}$, is a path terminating at zero. If we define

$$G(x) := \bigcup_{n=1}^{\lfloor \frac{-x}{3} \rfloor} G_n ,$$

then we have that $\{0, 1, x\}$ is a NADS if and only if $G(x)$ is a directed tree rooted at zero. If $\{0, 1, x\}$ is not a NADS then $G(x)$ must contain a directed cycle. In this section we discuss some of the properties of $G(x)$; in particular, we give a correspondence between strings in $\{00, 01, 0, 0x\}^*$ which represent nonzero multiples of Mersenne numbers and directed cycles of $G(x)$.

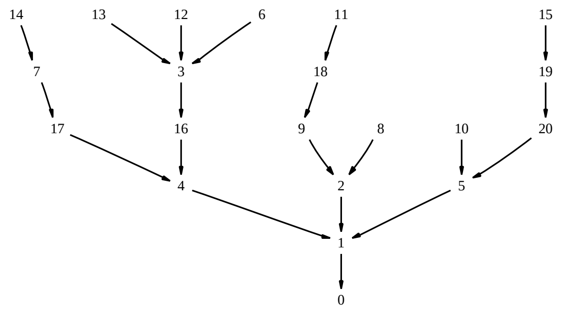

Figure 5.1: $G(-61)$

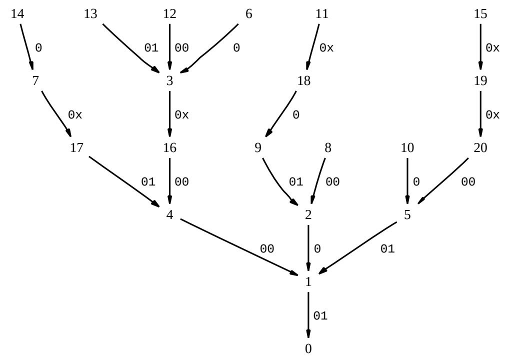

Figure 5.2: $G(-61)$ with arc labels.

We start with an example. Let $x = -61$. Since $\left\lfloor \frac{-x}{3} \right\rfloor = 20$, $G(x)$ is the union of $G_1, G_2, \ldots G_{20}$. A drawing of $G(x)$ is given in Figure 5.1. In Appendix B, it is noted that $\{0, 1, -61\}$ is a NADS and from Figure 5.1 we see that is indeed the case since $G(x)$ contains no directed cycle.

The function g_D, which was defined in (5.2), can be used to label the arcs of each of $G_1, G_2, \ldots G_{20}$ as follows:

$$n \xrightarrow{g_D(n)} f_D(n) \xrightarrow{g_D(f_D(n))} {f_D}^2(n) \xrightarrow{g_D({f_D}^2(n))} {f_D}^3(n) \xrightarrow{g_D({f_D}^3(n))} \cdots .$$

Recall that g_D returns a string from the set $\{00, 01, 0, 0x\}$. These arc labels can be applied to $G(x)$, as shown in Figure 5.2.

The arc labels on this drawing of $G(x)$ allow us to easily determine the D-NAF of any node of $G(x)$. If n is a node then, since $G(x)$ is a tree, there is a unique directed path from n to the root node (i.e., G_n). The sequence of arc labels on the reverse of this path identifies the $\{0,1,x\}$-NAF for n. For example, if we let $n = 14$, then from Figure 5.2 the directed path from 14 to 0 is

$$14 \xrightarrow{0} 7 \xrightarrow{0x} 17 \xrightarrow{01} 4 \xrightarrow{00} 1 \xrightarrow{01} 0\,.$$

If we read the sequence of arc labels above from right to left and concatenate them we get the string $01\|00\|01\|0x\|0$. It is easily verified that $14 = (0100010x0)_2$.

To see why this is true in general, suppose the path from n to 0 has length t and consider the label $g_D(n)$ on the arc $(n, f_D(n))$. From the definition of f_D and g_D we have

$$\begin{aligned} f_D(n) &= \frac{n - (g_D(n))_2}{2^{|g_D(n)|}} \\ \Longrightarrow \qquad n &= 2^{|g_D(n)|} f_D(n) + (g_D(n))_2\,, \end{aligned} \tag{5.6}$$

where $|g_D(n)|$ denotes the length of the string $g_D(n)$. Replacing n by $f_D(n)$ in (5.6) we have

$$f_D(n) = 2^{|g_D(f_D(n))|} {f_D}^2(n) + (g_D(f_D(n)))_2\,. \tag{5.7}$$

Substituting (5.7) into (5.6) we find

$$\begin{aligned} n &= 2^{|g_D(f_D(n))|+|g_D(n)|} {f_D}^2(n) + 2^{|g_D(n)|}(g_D(f_D(n)))_2 + (g_D(n))_2 \\ \Longrightarrow\ n &= 2^{|g_D(f_D(n))\|g_D(n)|} {f_D}^2(n) + (g_D(f_D(n))\|g_D(n))_2\,. \end{aligned}$$

This method of substitution can be applied again. In (5.6), n can be replaced by ${f_D}^2(n)$ and then we can use this new equation to substitute for ${f_D}^2(n)$ above, and so on.

Let α be the string formed by concatenating the arc labels along the reverse of the path from n to 0. Then we have:

$$\alpha = g_D({f_D}^{t-1}(n))\| \cdots \|g_D({f_D}^2(n))\|g_D(f_D(n))\|g_D(n)\,.$$

From (5.6), it follows that

$$n = 2^{|\alpha|} {f_D}^t(n) + (\alpha)_2. \tag{5.8}$$

Since the length of the path from n to 0 is t, ${f_D}^t(n) = 0$, and thus

$$n = (\alpha)_2 \,,$$

that is, α is a D-NAF for n.

The main result of this section concerns directed cycles in $G(x)$, so let us consider an example that contains a directed cycle. Let $x = -41$. This value of x is not listed in Appendix B, so we expect that $\{0, 1, -41\}$ is not a NADS, and the drawing of $G(x)$ in Figure 5.3 establishes this. Note, $G(x)$

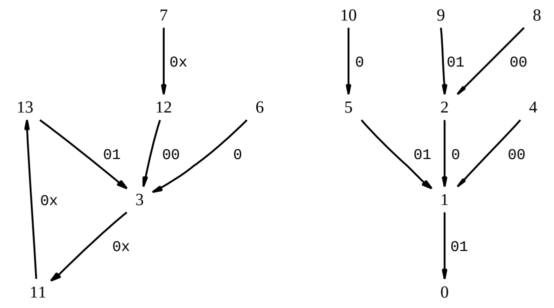

Figure 5.3: $G(-41)$ with arc labels.

consists of two components. Any node in the component of $G(x)$ which does not contain 0 does not have a D-NAF since there is no directed path from that node to 0.

Consider the directed cycle of $G(x)$. This cycle can be considered as a directed path from 3 to itself:

$$3 \xrightarrow{0x} 11 \xrightarrow{0x} 13 \xrightarrow{01} 3 \,.$$

Reading the sequence of arc labels above from right to left and concatenating them we get the string $01 \| 0x \| 0x$. This string has length 6 and because of this we claim that $2^6 - 1$ must divide $(010x0x)_2$. Since $x = -41$,

$(010x0x)_2 = -189$ and it is easy to check that this claim is valid. The following result provides an explanation.

Theorem 5.15. *Suppose x is a negative integer and $x \equiv 3 \pmod 4$. Then, $G(x)$ has a directed cycle if and only if $\exists \alpha \in \{00, 01, 0, 0x\}^*$ such that $(\alpha)_2 \neq 0$ and $2^{|\alpha|} - 1 \mid (\alpha)_2$.*

Proof. Suppose $G(x)$ has a directed cycle. Choose a node n in some directed cycle of $G(x)$ and let t be the length of this cycle. Then we have

$$n \xrightarrow{g_D(n)} f_D(n) \xrightarrow{g_D(f_D(n))} {f_D}^2(n) \to \cdots \to {f_D}^{t-1}(n) \xrightarrow{g_D({f_D}^{t-1}(n))} n \,.$$

Some node in this cycle must be congruent to 3 modulo 4. If not, then the iterates of f_D are strictly decreasing on this cycle and we get

$$n > f_D(n) > {f_D}^2(n) > \cdots > {f_D}^{t-1}(n) > n \,,$$

which is a contradiction. A consequence of this fact is that one of the arcs in the cycle is labeled $0x$. As before, let

$$\alpha = g_D({f_D}^{t-1}(n)) \| \cdots \| g_D({f_D}^2(n)) \| g_D(f_D(n)) \| g_D(n) \,.$$

Note, $(\alpha)_2 \neq 0$ because α contains the substring $0x$. Equation (5.8) gives us

$$n = 2^{|\alpha|} {f_D}^t(n) + (\alpha)_2 \,.$$

Since ${f_D}^t(n) = n$, we have

$$\begin{aligned} n &= 2^{|\alpha|} n + (\alpha)_2 \\ \implies -n(2^{|\alpha|} - 1) &= (\alpha)_2 \,. \end{aligned}$$

Thus, $(\alpha)_2 \neq 0$ and $2^{|\alpha|} - 1 \mid (\alpha)_2$, as required.

Suppose $\alpha \in \{00, 01, 0, 0x\}^*$ has the property that $(\alpha)_2 \neq 0$ and $2^{|\alpha|} - 1 \mid (\alpha)_2$. The string $0x$ must be a substring of α; otherwise, $0 < (\alpha)_2 < 2^{|\alpha|} - 1$, and this contradicts our hypothesis that $2^{|\alpha|} - 1 \mid (\alpha)_2$. We claim that we can assume $(\alpha)_2$ is odd. To see why, let α' be any left cyclic shift of α. For some $u \in \mathbb{Z}^+$, we have

$$\begin{aligned} &(\alpha')_2 \equiv 2^u (\alpha)_2 \pmod{2^{|\alpha|} - 1} \\ \implies &(\alpha')_2 \equiv 0 \pmod{2^{|\alpha|} - 1} , \end{aligned}$$

and since $|\alpha| = |\alpha'|$, this gives us that $2^{|\alpha'|} - 1 \mid (\alpha')_2$. Also, $(\alpha')_2 \neq 0$ because $(\alpha)_2 \neq 0$. Now, α contains the substring $0x$, so it must have some left cyclic shift that ends in 1 or x; that is, for some α', $(\alpha')_2$ is odd. Thus, if $(\alpha)_2$ is not odd, we can replace α by α' where $(\alpha')_2$ is odd.

Let $n = -\frac{(\alpha)_2}{2^{|\alpha|}-1}$. We will show n is in a directed cycle of $G(x)$. Since α contains the substring $0x$, $|\alpha| \geq 2$, and so we have the following:

$$\begin{aligned} &-n(2^{|\alpha|} - 1) = (\alpha)_2 \\ \implies \quad & n = 2^{|\alpha|} n + (\alpha)_2 \qquad (5.9) \\ \implies \quad & n \equiv (\alpha)_2 \pmod 4 \\ \implies \quad & \alpha = \alpha_1 \| g_D(n), \text{ where } \alpha_1 \in \{00, 01, 0, 0x\}^* . \end{aligned}$$

Using these implications, we can compute $f_D(n)$ as follows:

$$\begin{aligned} f_D(n) &= \frac{n - (g_D(n))_2}{2^{|g_D(n)|}} \\ &= \frac{2^{|\alpha|} n + (\alpha)_2 - (g_D(n))_2}{2^{|g_D(n)|}} \\ &= \frac{2^{|\alpha|} n + (\alpha_1 \| g_D(n))_2 - (g_D(n))_2}{2^{|g_D(n)|}} \\ &= 2^{|\alpha| - |g_D(n)|} n + (\alpha_1)_2 \\ &= 2^{|\alpha_1|} n + (\alpha_1)_2 . \qquad (5.10) \end{aligned}$$

Equation (5.10) is similar to equation (5.9). If $|\alpha_1| \geq 2$, the preceding arguments can be reapplied to compute ${f_D}^2(n)$. In doing so, we find

$${f_D}^2(n) = 2^{|\alpha_2|} n + (\alpha_2)_2 ,$$

where $\alpha_1 = \alpha_2 \| g_D(f_D(n))$ and $\alpha_2 \in \{00, 01, 0, 0x\}^*$. We can continue computing iterates of f_D in this manner until, for some $t \geq 1$, we obtain

$${f_D}^t(n) = 2^{|\alpha_t|} n + (\alpha_t)_2 ,$$

where $\alpha_{t-1} = \alpha_t \| g_D({f_D}^{t-1}(n))$, $\alpha_t \in \{00, 01, 0, 0x\}^*$ and $|\alpha_t| < 2$.

There are two cases to consider. If $|\alpha_t| = 0$ then it must be that $\alpha_t = \epsilon$, and thus

$${f_D}^t(n) = 2^0 n + (\epsilon)_2 = n\ .$$

Thus, n is in a directed cycle (of length t) in $G(x)$. If $|\alpha_t| = 1$ then it must be that $\alpha_t = 0$, and thus

$${f_D}^t(n) = 2^1 n + (0)_2 = 2n\ .$$

Recall that $(\alpha)_2$ is odd. Since $n = 2^{|\alpha|} n + (\alpha)_2$ and $|\alpha| \geq 2$, n is also odd. Thus, $2n \equiv 2 \pmod 4$ and so

$${f_D}^{t+1}(n) = \frac{2n}{2} = n\ .$$

Thus, n is in a directed cycle (of length $t+1$) in $G(x)$. □

Theorem 5.15 gives a complete characterization of NADS, however, it is unclear if this characterization is helpful in finding values of x which make $\{0, 1, x\}$ a NADS. On the other hand, Theorem 5.15 is very useful for finding values of x for which $\{0, 1, x\}$ is not a NADS. We give some examples of this in the next section.

One operation we used in the proof of Theorem 5.15 was a *cyclic shift* of the string $\alpha \in \{00, 01, 0, 0x\}^*$. If we consider this operation more closely, then we can strengthen Theorem 5.15.

For any nonempty string, consider the smallest positive number of cyclic shifts that when applied to this string results in the same string. Such an integer always exists since, if the string has length ℓ, then ℓ cyclic shifts will suffice. We call this number the *cyclic order* of the string. Using the quotient-remainder theorem, it is easy to show that the cyclic order of a string must divide its length.

Corollary 5.16. *In the statement of Theorem 5.15, we can impose the additional restriction that the string $\alpha \in \{00, 01, 0, 0x\}^*$ must have cyclic order equal to $|\alpha|$.*

Proof. We need to show that

i) $\exists \alpha \in \{00, 01, 0, 0x\}^*$ such that $(\alpha)_2 \neq 0$ and $2^{|\alpha|} - 1 \mid (\alpha)_2$, if and only if

ii) $\exists \alpha \in \{00, 01, 0, 0x\}^*$ such that $(\alpha)_2 \neq 0$, $2^{|\alpha|} - 1 \mid (\alpha)_2$ and the cyclic order of α equals $|\alpha|$.

Clearly, *ii*) implies *i*). To see that *i*) implies *ii*), suppose $\alpha \in \{00, 01, 0, 0x\}^*$ with $(\alpha)_2 \neq 0$ and $2^{|\alpha|} - 1 \mid (\alpha)_2$, but the cyclic order of α is less than $|\alpha|$. Let d be the cyclic order of α. Since d divides $|\alpha|$, we have

$$\alpha = \alpha_1 \| \alpha_1 \| \cdots \| \alpha_1$$

where α_1 is composed of the rightmost d digits of α, and the cyclic order of α_1 is equal to $|\alpha_1| = d$. Observe that $\frac{2^{|\alpha|}-1}{2^d-1}$ is an integer and

$$(\alpha)_2 = \frac{2^{|\alpha|} - 1}{2^d - 1}(\alpha_1)_2.$$

Now, since $(\alpha)_2 \neq 0$ we have $(\alpha_1)_2 \neq 0$. Also,

$$\begin{aligned} 2^{|\alpha|} - 1 \mid (\alpha)_2 &\implies \frac{2^{|\alpha|} - 1}{2^d - 1}(2^d - 1) \mid \frac{2^{|\alpha|} - 1}{2^d - 1}(\alpha_1)_2 \\ &\implies 2^d - 1 \mid (\alpha_1)_2. \end{aligned}$$

Since $d = |\alpha_1|$, we see that the string α_1 satisfies all the conditions of *ii*). □

The remainder of this chapter reads as follows. In Section 5.6, we give some infinite families of values for x for which D is *not* a NADS. In Section 5.7, we give some infinite families of values for x for which D is a NADS. We conclude by mentioning some additional problems related to NADS in Section 5.8.

5.6 Infinite Families of non-NADS

Consider the list of x values which appears in Appendix B. If we examine the first few entries of this list we find no multiples of 3. In fact, this is true of the whole list, and the same can be said of multiples of 7 and 31. These observations are a consequence of the following result:

Corollary 5.17. *Let x be a negative integer with $x \equiv 3 \pmod 4$. If $(2^s - 1)|x$ for any $s \geq 2$, then $\{0, 1, x\}$ is not a NADS.*

Proof. This result follows from Theorem 5.15, however it is just as easy to give a direct proof. Let $n = -x/(2^s - 1)$. We show G_n contains a directed cycle. We have

$$\begin{aligned} & n(2^s - 1) \equiv -x \pmod 4 \\ \Longrightarrow\ & n(0 - 1) \equiv -3 \pmod 4 \\ \Longrightarrow\ & n \equiv 3 \pmod 4 . \end{aligned}$$

Note that,

$$n - x = \frac{-x}{2^s - 1} - x = \frac{-x - x(2^s - 1)}{2^s - 1} = 2^s \frac{-x}{2^s - 1} = 2^s n .$$

Now,

$$f_D(n) = \frac{n - x}{4} = 2^{s-2} n$$

Subsequent iterates of f_D will cancel out the factor 2^{s-2}. Thus, for some i, ${f_D}^i(n) = n$ and so G_n contains a directed cycle. □

Corollary 5.17 says that many sets $\{0, 1, x\}$ are not NADS. In particular, it rules out sets where x is divisible by 3, 7, 31, etc. Besides numbers of the form $2^s - 1, s \geq 2$, there are many other *non-allowable factors* of x. For example, if any of the integers

$$73, 85, 89, 337, 451, 1103, 1205, 1285, 2089$$

divides x then it is possible to show that, for a carefully chosen value of n, G_n contains a directed cycle. This technique of proof is not fully satisfying since it does little to elucidate why one integer is a non-allowable factor and another is not. A better approach is presented in the following corollary to Theorem 5.15.

Corollary 5.18. *Suppose x_0 is an integer. If $\exists \beta \in \{00, 0, 0x_0\}^*$ such that $(\beta)_2 \neq 0$ and $2^{|\beta|} - 1 \mid (\beta)_2$ then x_0 is a non-allowable factor.*

Proof. Notice there are no restrictions put on the integer x_0. Let x be a negative integer with $x \equiv 3 \pmod 4$ and $x_0 | x$. We must show $\{0, 1, x\}$ is not a NADS. Let α be the string formed by changing every occurrence of x_0 in β to x. It is easy to see that $(\alpha)_2 = \frac{x}{x_0}(\beta)_2$, $\alpha \in \{00, 0, 0x\}^*$ and $|\alpha| = |\beta|$. Now,

$$\begin{aligned} & 2^{|\beta|} - 1 \mid (\beta)_2 \\ \Longrightarrow\ & 2^{|\beta|} - 1 \mid \frac{x}{x_0}(\beta)_2 \\ \Longrightarrow\ & 2^{|\beta|} - 1 \mid (\alpha)_2 \\ \Longrightarrow\ & 2^{|\alpha|} - 1 \mid (\alpha)_2 \end{aligned}$$

Since $\alpha \in \{00, 01, 0, 0x\}^*$ and $(\alpha)_2 \neq 0$, by Theorem 5.15 we have that $\{0, 1, x\}$ is not a NADS. ☐

We can use this result to generate non-allowable factors. All we need to do is find an integer x_0 and a string $\beta \in \{00, 0, 0x_0\}^*$, where β is not an all-zero string, such that $2^{|\beta|} - 1 \mid (\beta)_2$. To do this we first choose a string $\beta' \in \{00, 0, 01\}^*$ that is not an all-zero string. Now, we find an integer x_0 such that $2^{|\beta'|} - 1 \mid x_0(\beta')_2$. The smallest positive value of x_0 that satisfies this relation is

$$\frac{2^{|\beta'|} - 1}{\gcd(2^{|\beta'|} - 1, (\beta')_2)} .$$

We assign x_0 this value. If we change each occurrence of 1 in the string β' to x_0 we get a string $\beta \in \{00, 0, 0x_0\}^*$ such that $(\beta)_2 \neq 0$ and $2^{|\beta|} - 1 \mid (\beta)_2$. So, by the corollary, x_0 is a non-allowable factor. Here is a short example. Let $\beta' = 000010101$. Then $|\beta'| = 9$, $(\beta')_2 = 21$, and so

$$x_0 = \frac{2^9 - 1}{\gcd(2^9 - 1, 21)} = 73 .$$

Thus, 73 is a non-allowable factor.

More generally, Theorem 5.15 be can used to generate infinite families of non-NADS which do not necessarily involve non-allowable factors. We know $\{0, 1, x\}$ is not an NADS if we can find a string $\alpha \in \{00, 01, 0, 0x\}^*$ such that $-n(2^{|\alpha|} - 1) = (\alpha)_2$. If we fix α and solve the resulting integer equation for x this will give us an infinite family of non-NADS. For example, suppose we fix $\alpha = 01010x0x$, then

$$\begin{aligned} & -n(2^{|\alpha|}-1) = (01010x0x)_2 \\ \iff & -n(2^8-1) = (01010000)_2 + x(00000101)_2 \\ \iff & -255n = 80 + 5x \\ \iff & -51n = 16 + x\ . \end{aligned}$$

Thus, if $x \equiv -16 \pmod{51}$ then $\{0, 1, x\}$ cannot be a NADS.

Some of our first results on infinite families of non-NADS, which were discovered empirically, are unified as corollaries of Theorem 5.15. The following two results demonstrate this.

Corollary 5.19. *If $\frac{3-x}{4} = 11 \cdot 2^i$, where $i \geq 0$, then $\{0, 1, x\}$ is not a NADS.*

Proof. We have,

$$\begin{aligned} & \frac{3-x}{4} = 11 \cdot 2^i \\ \implies & 3 - x = 11 \cdot 2^{i+2} \\ \implies & 11 - x = 11 \cdot 2^{i+2} + 8 \\ \implies & -11(2^{i+2}-1) = 8 + x \\ \implies & -11(2^{i+2}-1) = (0100x)_2\ . \end{aligned}$$

The length of the string $0100x$ is 5. If $i + 2 \geq 5$ we can prepend zeros to $0100x$ and build a string α such that $|\alpha| = i + 2$; thus, by Theorem 5.15 we are done. If $i + 2 < 5$, it must be that $i = 0, 1, 2$.

When $i = 0$, $x = -41$ and from the drawing in Figure 5.3 we see $G(-41)$ has a directed cycle. When $i = 1$, $x = -85$ and then G_3 is a directed cycle:

$$3 \to 22 \to 11 \to 24 \to 6 \to 3\ .$$

When $i = 2$, $x = -173$ and G_3 is also a directed cycle:

$$3 \to 44 \to 11 \to 46 \to 23 \to 49 \to 12 \to 3\ .$$

In any case, $\{0, 1, x\}$ is not an NADS, as required. □

Corollary 5.20. *Let $\frac{3-x}{4} = 7 \cdot 2^i$, where $i \geq 0$. Then $\{0, 1, x\}$ is an NADS if and only if $i \in \{0, 1\}$.*

Proof. We have,

$$\begin{aligned} &\frac{3-x}{4} = 7 \cdot 2^i \\ \Longrightarrow\ &3 - x = 7 \cdot 2^{i+2} \\ \Longrightarrow\ &7 - x = 7 \cdot 2^{i+2} + 4 \\ \Longrightarrow\ &-7(2^{i+2} - 1) = 4 + x \\ \Longrightarrow\ &-7(2^{i+2} - 1) = (010x)_2\ . \end{aligned}$$

Arguing as in the previous corollary, if $i + 2 \geq 4$ then by Theorem 5.15, $\{0, 1, x\}$ is not a NADS. If $i + 2 < 4$, it must be that $i = 0, 1$.

When $i = 0$, $x = -25$ and when $i = 1$, $x = -53$. By drawing the graphs $G(-25)$ and $G(-53)$, it is easy to verify that both of these values give NADSs (this is confirmed in Appendix B). □

Not all infinite families of non-NADS are derived from Theorem 5.15. Consider the set of integers $\text{NAF}(\{0, 1\})$. If this set is ordered, from smallest to largest, we sometimes notice large gaps between consecutive elements. One type of gap is described as follows. For $i \geq 0$, let

$$m_i := \left\lfloor \frac{2^{i+1} - 1}{3} \right\rfloor .$$

Computing the first few values of m_i, we have

i	m_i
0	0
1	$1 = (1)_2$
2	$2 = (10)_2$
3	$5 = (101)_2$
4	$10 = (1010)_2$
5	$21 = (10101)_2$
6	$42 = (101010)_2$
7	$85 = (1010101)_2$
$\vdots$	$\vdots$

It is easy to see that if $a \in \text{NAF}(\{0,1\})$ then it is never true that $m_i < a < 2^i$. This observation gives us another infinite family.

Theorem 5.21. *Let x be an integer such that $4m_i - 1 < -x < 3 \cdot 2^i$ for some $i \geq 0$. If there exists $n \in \{1, 2, \ldots \lfloor -x/3 \rfloor\}$ with $n \equiv 3 \pmod 4$ then $\{0, 1, x\}$ is not a NADS.*

Proof. We can assume $x \equiv 3 \pmod 4$, since otherwise $\{0, 1, x\}$ cannot be a NADS. Suppose to the contrary that $\{0, 1, x\}$ is a NADS. Then, in the graph $G(x)$, there must be a directed path from n to 0. Let n_0 be the integer on this path that is *closest* to 0 and is congruent to 3 modulo 4. The arc labels on the path from n_0 to 0 give the $\{0, 1, x\}$-NAF for n_0. It must be that $n_0 = (\alpha \| 0x)_2$ with $\alpha \in \{00, 01, 0\}^*$ (if α contained the substring $0x$ this would contradict our choice of n_0).

Now,

$$\begin{aligned}
& 1 \leq n_0 \leq -x/3 \\
\Longrightarrow\ & 1 \leq (\alpha \| 0x)_2 \leq -x/3 \\
\Longrightarrow\ & 1 \leq 4(\alpha)_2 + x \leq -x/3 \\
\Longrightarrow\ & \frac{1-x}{4} \leq (\alpha)_2 \leq \frac{-x/3 - x}{4} \\
\Longrightarrow\ & \frac{1-x}{4} \leq (\alpha)_2 \leq -x/3\,.
\end{aligned}$$

By hypothesis, we have

$$4m_i - 1 < -x \quad \text{and} \quad -x < 3 \cdot 2^i$$
$$\Longrightarrow m_i < \frac{1-x}{4} \quad \text{and} \quad \frac{-x}{3} < 2^i$$

Thus, for some $i \geq 0$, we have

$$m_i < (\alpha)_2 < 2^i$$

which is a contradiction. Thus, $\{0, 1, x\}$ is not a NADS. □

For example, if $i = 5$ then $-(4m_5 - 1) = -83$ and $-3 \cdot 2^5 = -96$. Theorem 5.21 tells us that no value of x with $-83 < x < -96$ can give a NADS. In addition, the proof of Theorem 5.21 also gives us some information about the graphs $G(x)$ for such values of x. For each of these graphs, in the component that contains 0 there can be no integer congruent to 3 modulo 4 (or equivalently, no arc label in this component can be $0x$). This property can be observed in $G(-85)$ which is drawn in Figure 5.4. Avoine, Monnerat and Peyrin [3] have observed that the converse of this statement also holds.

Proposition 5.22 (Avoine, Monnerat, Peyrin). *If $G(x)$ has the property that no arc in the component of $G(x)$ that contains 0 is labeled $0x$, then x must satisfy $4m_i - 1 < -x < 3 \cdot 2^i$ for some $i \geq 0$.*

A proof of this fact can be found in [3], however, we provide a different proof here.

Proof. We prove the contrapositive of the statement. Suppose x does not satisfy $4m_i - 1 < -x < 3 \cdot 2^i$ for any $i \geq 0$. Define

$$i_0 := \lfloor \lg(-x/3) \rfloor,$$

and observe that

$$3 \cdot 2^{i_0} \leq -x \leq 4m_{i_0+1} - 1.$$

The value 2^{i_0+2} is in this interval so either

$$3 \cdot 2^{i_0} \leq -x < 2^{i_0+2} \qquad \text{or} \qquad 2^{i_0+2} \leq -x \leq 4m_{i_0+1} - 1.$$

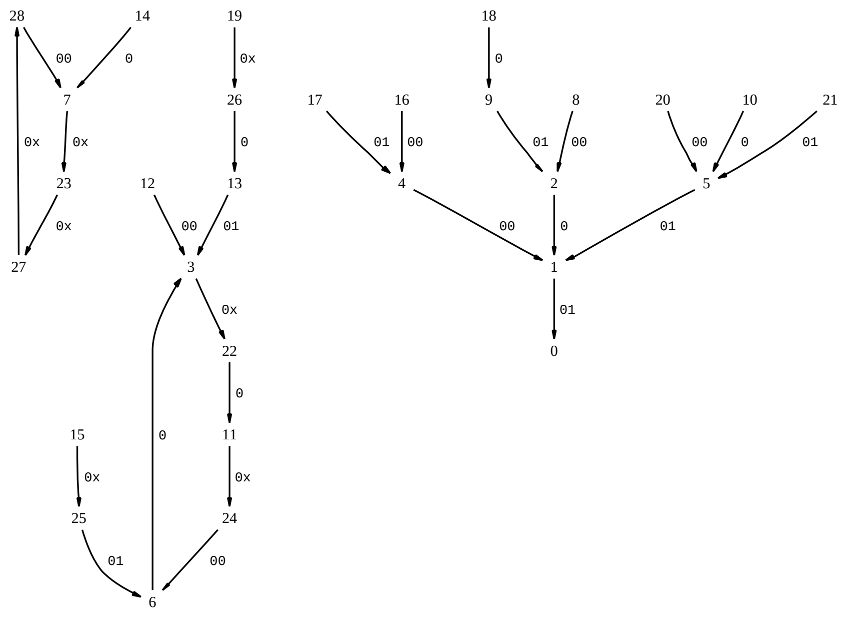

Figure 5.4: $G(-85)$ with arc labels.

If the bound on the left holds, consider $n = 2^{i_0+2} + x$. Clearly, n has a $\{0, 1, x\}$-NAF. Note that

$$\begin{aligned} -x &< 2^{i_0+2} \\ \implies 0 &< 2^{i_0+2} + x \\ \implies 1 &\leq n \end{aligned} \qquad\qquad \begin{aligned} 3 \cdot 2^{i_0} &\leq -x \\ \implies 2^{i_0+2} &\leq -4x/3 \\ \implies 2^{i_0+2} + x &\leq -x/3 \\ \implies n &\leq -x/3. \end{aligned}$$

So, n is a vertex of $G(x)$, it is in the same component as 0 and the arc leaving it is labeled $0x$.

If the bound on the right holds, consider $n = 4m_{i_0+1} + x$. Clearly, n has a $\{0, 1, x\}$-NAF. Note that

$$\begin{aligned} -x &\leq 4m_{i_0+1} - 1 \\ \implies 1 &\leq 4m_{i_0+1} + x \\ \implies 1 &\leq n \end{aligned} \qquad \begin{aligned} 2^{i_0+2} &\leq -x \\ \implies (2^{i_0+2} - 1)/3 &< -x/3 \\ \implies m_{i_0+1} &< -x/3 \\ \implies 4m_{i_0+1} + x &< -4x/3 + x \\ \implies n &< -x/3. \end{aligned}$$

So, n is a vertex of $G(x)$, it is in the same component as 0 and the arc leaving it is labeled $0x$.

This proves the result. □

5.7 Infinite Families of NADS

If n is a nonnegative integer, $w(n)$ denotes the number of ones in the usual $\{0, 1\}$-radix 2 representation of n (i.e., the Hamming weight of n). We use the function $w(n)$ to describe two infinite families.

Theorem 5.23. *Let x be a negative integer with $x \equiv 3 \pmod 4$. If $w\left(\frac{3-x}{4}\right) = 1$, then $\{0, 1, x\}$ is a NADS.*

Proof. Suppose $\{0, 1, x\}$ is not a NADS. Then there is some $n \in \mathbb{Z}^+$ for which the graph G_n contains a directed cycle. We can assume n is a vertex of this cycle. Let t be the number of vertices in the cycle, then

$$n \to f_D(n) \to {f_D}^2(n) \to \cdots \to {f_D}^{t-1}(n) \to n\ .$$

Let $n' = f_D(n)$. We want to relate $w(n')$ to $w(n)$. There are four possible residues of n modulo 4, and for the residues $0, 1, 2$ we can determine $w(n')$ exactly:

$n \bmod 4$	n'	$w(n')$
0	$\frac{n}{4}$	$w(n)$
1	$\frac{n-1}{4}$	$w(n) - 1$
2	$\frac{n}{2}$	$w(n)$

If $n \equiv 3 \pmod 4$, we have

$$n' = \frac{n-x}{4} = \frac{n-3}{4} + \frac{3-x}{4} \, .$$

By hypothesis $w\left(\frac{3-x}{4}\right) = 1$, and so

$$\begin{aligned} w(n') &= w\left(\frac{n-3}{4} + \frac{3-x}{4}\right) \\ &\le w\left(\frac{n-3}{4}\right) + w\left(\frac{3-x}{4}\right) \\ &= w(n) - 2 + 1 \\ &= w(n) - 1 \, . \end{aligned}$$

So, in any case, $w(n') \le w(n)$, but if n is odd then we have the strict inequality $w(n') < w(n)$. Applying this inequality to the integers in the cycle of G_n we see

$$w(n) \ge w(f_D(n)) \ge w({f_D}^2(n)) \ge \cdots \ge w({f_D}^{t-1}(n)) \ge w(n) \, .$$

However, some vertex in this cycle must be congruent to 3 modulo 4. If not, then the iterates of f_D are strictly decreasing on this cycle and we get

$$n > f_D(n) > {f_D}^2(n) > \cdots > {f_D}^{t-1}(n) > n \, ,$$

which is a contradiction. So, there is some odd vertex in the cycle which means one of the inequalities relating the Hamming weights of adjacent vertices is strict. This implies that $w(n) > w(n)$, which is a contradiction.

So, G_n cannot contain a directed cycle, and hence $\{0, 1, x\}$ is a NADS. □

When x is negative, $w(\frac{3-x}{4}) = 1$ if and only if $\frac{3-x}{4} = 2^t$, $t \ge 0$. Letting $t = 0, 1, 2, 3, 4, \ldots$ we see that Theorem 5.23 asserts that $x = -1, -5, -13, -29, -61, \ldots$ all yield NADS. Our next result also describes an infinite family using the function $w(n)$. However, when compared to the previous result, proving that $\{0, 1, x\}$ is a NADS for each x in this second infinite family seems to be more difficult.

Theorem 5.24. *Let x be a negative integer with $x \equiv 3 \pmod 4$. If $w\left(\frac{3-x}{4}\right) = 2$ and $2^s - 1$ does not divide x for any $s \in \mathbb{Z}^+$, $s \geq 2$, then $\{0, 1, x\}$ is a NADS.*

To prove this result we suppose x is a negative integer with $x \equiv 3 \pmod 4$ and $w\left(\frac{3-x}{4}\right) = 2$. We will argue that if $\{0, 1, x\}$ is not a NADS then it must be that $2^s - 1$ divides x for some $s \in \mathbb{Z}^+$, $s \geq 2$.

Suppose $\{0, 1, x\}$ is not a NADS. Then, by Theorem 5.15 there exists a string $\alpha \in \{00, 01, 0, 0x\}^*$ such that $(\alpha)_2 \neq 0$ and $2^{|\alpha|} - 1 \mid (\alpha)_2$. Moreover, by the proof of Theorem 5.15, if we take $n = \frac{-(\alpha)_2}{2^{|\alpha|}-1}$, then G_n is a directed cycle and the concatenation of its arc labels gives the string α.

Let t be the number of vertices in G_n. Then we have

$$n \xrightarrow{g_D(n)} f_D(n) \xrightarrow{g_D(f_D(n))} {f_D}^2(n) \to \cdots \to {f_D}^{t-1}(n) \xrightarrow{g_D({f_D}^{t-1}(n))} n \, .$$

Let $n' = f_D(n)$. We want to relate $w(n')$ to $w(n)$. There are four possible residues of n modulo 4, and for the residues $0, 1, 2$ we can determine $w(n')$ exactly:

$n \bmod 4$	n'	$w(n')$
0	$\frac{n}{4}$	$w(n)$
1	$\frac{n-1}{4}$	$w(n) - 1$
2	$\frac{n}{2}$	$w(n)$

If $n \equiv 3 \pmod 4$, we have

$$n' = \frac{n-x}{4} = \frac{n-3}{4} + \frac{3-x}{4} \, .$$

By hypothesis $w\left(\frac{3-x}{4}\right) = 2$, and so

$$\begin{aligned} w(n') &= w\left(\frac{n-3}{4} + \frac{3-x}{4}\right) \\ &\leq w\left(\frac{n-3}{4}\right) + w\left(\frac{3-x}{4}\right) \\ &= w(n) - 2 + 2 \\ &= w(n) \, . \end{aligned}$$

So, in any case, $w(n') \leq w(n)$, but if $n \equiv 1 \pmod 4$ then we have the strict inequality $w(n') < w(n)$. Applying this inequality to the integers in the cycle of G_n we see

$$w(n) \geq w(f_D(n)) \geq w({f_D}^2(n)) \geq \cdots \geq w({f_D}^{t-1}(n)) \geq w(n)\,.$$

No vertex in this cycle can be congruent to 1 modulo 4; otherwise, one of the inequalities above would be strict and this would imply $w(n) > w(n)$, which is a contradiction. Also, at least one vertex in this cycle is congruent to 3 modulo 4; otherwise, by definition of f_D, the vertices would form a strictly decreasing integer sequence which would imply $n > n$, which is a contradiction.

Since no arc in G_n is labeled 01, α is a concatenation of strings from the set $\{00, 0, 0x\}$. Note that α is nonadjacent, and every cyclic shift of α is also nonadjacent (i.e., α is *cyclically nonadjacent*).

The integer $(\alpha)_2$ is divisible by x. Let

$$A = \frac{(\alpha)_2}{x}, \quad \text{and} \quad a = |\alpha|\,.$$

Since $2^{|\alpha|} - 1 \mid (\alpha)_2$ we have

$$-xA \equiv 0 \pmod{2^a - 1}\,. \tag{5.11}$$

Since $w\left(\frac{3-x}{4}\right) = 2$, for some $u, v \in \mathbb{Z}$ we have

$$-x = 2^u + 2^v - 3, \quad u > v \geq 2\,,$$

and now (5.11) implies

$$(2^u + 2^v - 3)A \equiv 0 \pmod{2^a - 1}, \text{ where } u > v \geq 2\,. \tag{5.12}$$

To finish the proof we need a lemma. Before we can introduce the lemma, we need a definition.

Definition 5.25. An integer $B \in \mathbb{Z}$ is *length-ℓ cyclically nonadjacent* if $B \neq 0$ and there is a cyclically nonadjacent string $\beta \in \{0,1\}^\ell$ such that $(\beta)_2 = B$.

Note that in this definition, the string β may have leading zeros. For example, 21 is length-6 cyclically nonadjacent (6-CNA, for short) since the string $010101 \in \{0,1\}^6$ is cyclically nonadjacent and $(010101)_2 = 21$. However, 21 is not 5-CNA because the only string in $\{0,1\}^5$ which gives a representation of 21 is 10101, but the cyclic shift 01011 of this string is not nonadjacent. Now we are ready for the lemma.

Lemma 5.26. *If B is length-ℓ cyclically nonadjacent and the congruence*

$$(2^u + 2^v - 3)B \equiv 0 \pmod{2^\ell - 1}$$

holds for some $u, v \in \mathbb{Z}$, $u > v \geq 2$, then either

$$\gcd(u, v-1) > 1 \quad \textit{or} \quad \gcd(u-1, v) > 1 \, .$$

Assuming, for the moment, the truth of Lemma 5.26, our proof of Theorem 5.24 continues as follows. The string α is cyclically nonadjacent, therefore so is the string formed by changing each occurrence of x in α to 1. This establishes that A is length-a cyclically nonadjacent, because $A = \frac{(\alpha)_2}{x}$. Now we can apply Lemma 5.26 to (5.12) and deduce, without loss of generality, that $\gcd(u, v-1) > 1$. Let $s = \gcd(u, v-1)$. Note that

$$-x = 2^u + 2^v - 3 = (2^u - 1) + 2(2^{v-1} - 1) \, .$$

Since $\gcd(2^u - 1, 2^{v-1} - 1) = 2^{\gcd(u,v-1)} - 1 = 2^s - 1$ we have that $2^s - 1 \mid x$, where $s \in \mathbb{Z}^+$ and $s \geq 2$, which is exactly what we wanted to show. (If x was chosen so as to satisfy all the conditions of Theorem 5.24, $2^s - 1$ cannot divide x, thus it must be that $\{0, 1, x\}$ is a NADS.) This concludes our proof of Theorem 5.24 , however we still have to deal with Lemma 5.26.

In proving Lemma 5.26, we will make use of the following easy result:

Lemma 5.27. *For any two nonempty subsets $S, T \subseteq \{0, 1, \ldots, \ell - 1\}$,*

$$\sum_{s \in S} 2^s \equiv \sum_{t \in T} 2^t \pmod{2^\ell - 1}$$

if and only if $S = T$.

Proof. We have $0 < \sum_{s\in S} 2^s \leq 2^\ell - 1$, and similarly for $\sum_{t\in T} 2^t$. Thus,

$$\begin{aligned} & \sum_{s\in S} 2^s \equiv \sum_{t\in T} 2^t \pmod{2^\ell - 1} \\ \iff & \sum_{s\in S} 2^s = \sum_{t\in T} 2^t \\ \iff & S = T\,. \end{aligned}$$

□

Proof of Lemma 5.26. We fix some notation that will help describe our proof of Lemma 5.26. From now on, we let B be an integer which satisfies the hypothesis of Lemma 5.26. B is ℓ-CNA and we let $\beta = b_{\ell-1}\dots b_1 b_0$ be the string in $\{0,1\}^\ell$ which establishes this. Further, let $S = \{i : b_i = 1\}$. For $k \in \mathbb{Z}$, define

$$S + k = \{(s+k) \bmod \ell : s \in S\}\,.$$

The set $S + k$ is called a *translate* of S modulo ℓ. Using this notation, we have

$$\begin{aligned} & (2^u + 2^v - 3)B \equiv 0 \pmod{2^\ell - 1} \\ \iff & (2^u + 2^v)B \equiv 3B \pmod{2^\ell - 1} \\ \iff & (S+u) \cup (S+v) = (S+1) \cup S\,, \end{aligned} \tag{5.13}$$

where the last equivalence follows from the fact that B is ℓ-CNA and Lemma 22. Because B is ℓ-CNA, the union on the right-hand side of (5.13), and hence also the left-hand side, is disjoint. We will establish Lemma 5.26 by analyzing this set equality.

We need one more concept. We previously introduced the *cyclic order* of a string. We define $\overset{\leftrightarrow}{\mathrm{ord}}(B)$ to be the cyclic order of the string $\beta = b_{\ell-1}\dots b_1 b_0$ (recall that $B = (\beta)_2$). Equivalently, $\overset{\leftrightarrow}{\mathrm{ord}}(B)$ can be defined as the smallest positive integer k such that

$$2^k B \equiv B \pmod{2^\ell - 1}\,.$$

Such an integer always exists since

$$2^\ell B \equiv B \pmod{2^\ell - 1}\,.$$

Using the quotient-remainder theorem, it is easy to show for any $m \in \mathbb{Z}^+$ that

$$2^m B \equiv B \pmod{2^\ell - 1} \iff \overset{\hookrightarrow}{\mathrm{ord}}(B) | m \,.$$

Applying this result, we see that $\overset{\hookrightarrow}{\mathrm{ord}}(B)|\ell$.

We claim that we can assume $\overset{\hookrightarrow}{\mathrm{ord}}(B) = \ell$ in the hypotheses of Lemma 5.26. We justify this claim as follows. Let $k = \overset{\hookrightarrow}{\mathrm{ord}}(B)$ and suppose $k < \ell$. Since $k|\ell$ we can write $km = \ell$ for some positive integer m. Since B is ℓ-CNA we have

$$B = (2^{(m-1)k} + \cdots + 2^{2k} + 2^k + 1)B' = \frac{2^\ell - 1}{2^k - 1} B'$$

where $B' = (b_{k-1} \dots b_1 b_0)_2$, and B' is k-CNA. Now, for any positive integer j, we have

$$\begin{aligned} & 2^j B \equiv B \pmod{2^\ell - 1} \\ \iff\ & 2^j \frac{2^\ell - 1}{2^k - 1} B' \equiv \frac{2^\ell - 1}{2^k - 1} B' \pmod{\frac{2^\ell - 1}{2^k - 1}\, 2^k - 1} \\ \iff\ & 2^j B' \equiv B' \pmod{2^k - 1} \,, \end{aligned}$$

and so it must be that $\overset{\hookrightarrow}{\mathrm{ord}}(B') = k$ (i.e., $\overset{\hookrightarrow}{\mathrm{ord}}(B')$ is as large as possible). Also, we have

$$\begin{aligned} & (2^u + 2^v - 3)B \equiv 0 \pmod{2^\ell - 1} \\ \iff\ & (2^u + 2^v - 3)\, \frac{2^\ell - 1}{2^k - 1}\, B' \equiv 0 \pmod{\frac{2^\ell - 1}{2^k - 1}\, 2^k - 1} \\ \iff\ & (2^u + 2^v - 3)B' \equiv 0 \pmod{2^k - 1} \,. \end{aligned}$$

So if we can prove Lemma 5.26 for all B with $\overset{\hookrightarrow}{\mathrm{ord}}(B)$ as large as possible, then by the above arguments, it is true for all B.

Returning to the set equality described in (5.13), recall $S \subseteq \{0, 1, \dots, \ell - 1\}$. Since S is a subset of integers its elements can be ordered from smallest to largest. From S we define a sequence, $d(S)$, of differences modulo ℓ:

$$d(S) := (s_1 - s_0, s_2 - s_1, \dots, s_{p-1} - s_{p-2}, s_0 - s_{p-1})$$

where

$$S = \{s_0, s_1, \ldots, s_{p-1}\} \quad \text{with} \quad s_0 < s_1 < \cdots < s_{p-1} \, .$$

Because B is ℓ-CNA, each of the differences in the sequence $d(S)$ must be at least 2. The definition of $d(S)$ can be extended to the translates of S. For any $k \in \mathbb{Z}$, $S + k$ can be considered as a subset of $\{0, 1, \ldots, \ell - 1\}$ and hence it can also be ordered from smallest to largest. Thus, $d(S + k)$ can be defined in the same way as $d(S)$. It is easy to show that $d(S + k)$ is a cyclic shift of $d(S)$. Because of this property there are at most p different sequences of the form $d(S + k)$ where $p = |S|$. In fact, we can show there are exactly p such sequences.

Let

$$t_i := \ell - s_i \, , \text{ for } 0 \le i \le p - 1 \, .$$

The smallest element in each of the translates $S + t_0, S + t_1, \ldots, S + t_{p-1}$ is equal to 0. Thus, for $i, j \in \{0, 1, \ldots, p - 1\}$, we have

$$d(S + t_i) = d(S + t_j) \iff S + t_i = S + t_j \, .$$

Let $i \ge j$. Then we have

$$\begin{aligned}
& S + t_i = S + t_j \\
\iff \quad & 2^{t_i} B \equiv 2^{t_j} B \pmod{2^\ell - 1} \\
\iff \quad & 2^{t_i - t_j} B \equiv B \pmod{2^\ell - 1} \\
\iff \quad & \overset{\hookrightarrow}{\mathrm{ord}}(B) \mid (t_i - t_j) \\
\iff \quad & \ell \mid (t_i - t_j) \\
\iff \quad & t_i = t_j \, .
\end{aligned}$$

So, each of the sequences $d(S + t_0), d(S + t_1), \ldots, d(S + t_{p-1})$ is distinct and hence there are exactly p different sequences of the form $d(S + k)$ where $k \in \mathbb{Z}$.

By applying a lexicographical ordering to the sequences $d(S + t_0)$, $d(S + t_1), \ldots, d(S + t_{p-1})$ we can identify a *unique* smallest sequence. Let

t^* be the value of t_i which corresponds to this smallest sequence. Note that

$$\begin{aligned} & (S+u) \cup (S+v) = (S+1) \cup S \\ \iff & \big((S+u) \cup (S+v)\big) + t^* = \big((S+1) \cup S\big) + t^* \\ \iff & (S+u+t^*) \cup (S+v+t^*) = (S+1+t^*) \cup (S+t^*) \ . \end{aligned} \tag{5.14}$$

We have $0 \in S+t^*$, so either $0 \in S+u+t^*$ or $0 \in S+v+t^*$. Without loss of generality we can assume $0 \in S+v+t^*$. We will show $S+v+t^* = S+t^*$.

Let

$$d(S+t^*) = (d_0, d_1, d_2, \dots, d_{p-1}) \ ,$$

and note that

$$S+t^* = \{0, d_0, d_1+d_0, d_2+d_1+d_0, \dots\} \ .$$

Also, let

$$\begin{aligned} d(S+u+t^*) &= (u_0, u_1, u_2, \dots, u_{p-1}) \\ d(S+v+t^*) &= (v_0, v_1, v_2, \dots, v_{p-1}) \ . \end{aligned}$$

Since $d(S+t^*)$ is a lexicographically smallest sequence of the form $d(S+k)$ where $k \in \mathbb{Z}$, we have

$$d(S+t^*) \le d(S+u+t^*) \quad \text{and} \quad d(S+t^*) \le d(S+v+t^*) \ .$$

Recall $0 \in S+t^*$ and $0 \in S+v+t^*$. Since $0 \in S+t^*$, we have $1 \in S+1+t^*$. By (5.14), either $1 \in S+u+t^*$ or $1 \in S+v+t^*$. Suppose $1 \in S+v+t^*$. Then both 0 and 1 are elements of $S+v+t^*$. No two elements in any translate of S can have a difference of 1; otherwise, this contradicts the fact that B is ℓ-CNA. So, it must be that $1 \in S+u+t^*$.

We now know the smallest elements in each of the sets $S+u+t^*$, $S+v+t^*$, $S+1+t^*$, $S+t^*$. The next smallest element of $S+t^*$ is d_0. Again, by (5.14), either $d_0 \in S+u+t^*$ or $d_0 \in S+v+t^*$. Suppose $d_0 \in S+u+t^*$. Then, since both 1 and d_0 are in $S+u+t^*$ and 1 is the smallest element of this set, we have

$$u_0 \le d_0 - 1 < d_0 \ .$$

However, $d(S+t^*) \leq d(S+u+t^*)$ implies that $d_0 \leq u_0$ which gives a contradiction. So, it must be that $d_0 \in S+v+t^*$, and hence, $d_0+1 \in S+u+t^*$.

From our lexicographical ordering we have $d_0 \leq v_0$. Since the smallest element of $S+v+t^*$ is 0 and d_0 is also in this set, we have

$$v_0 \leq d_0 - 0 = d_0 \ .$$

Hence, $v_0 = d_0$. Similarly,

$$d_0 \leq u_0 \quad \text{and} \quad u_0 \leq (d_0+1)-1 = d_0 \ ,$$

and so $u_0 = d_0$. From these two equalities, we have that d_0 and d_0+1 are the second smallest elements of the sets $S+v+t^*$ and $S+u+t^*$, respectively. Further, our lexicographical ordering now implies that $d_1 \leq v_1$ and $d_1 \leq u_1$.

The next smallest element of $S+t^*$ is d_1+d_0. Either $d_1+d_0 \in S+u+t^*$ or $d_1+d_0 \in S+v+t^*$. Suppose $d_1+d_0 \in S+u+t^*$. This implies that

$$u_1 \leq (d_1+d_0)-(d_0+1) = d_1 - 1 < d_1 \ ,$$

which is a contradiction. So, $d_1+d_0 \in S+v+t^*$, and hence, $d_1+d_0+1 \in S+u+t^*$.

We now have

$$d_1 \leq v_1 \quad \text{and} \quad v_1 \leq (d_1+d_0+1)-(d_0+1) = d_1 \ ,$$

so $v_1 = d_1$. Also

$$d_1 \leq u_1 \quad \text{and} \quad u_1 \leq (d_1+d_0)-d_0 = d_1 \ ,$$

and so $u_1 = d_1$. Thus we can identify the third smallest elements of the sets $S+v+t^*$ and $S+u+t^*$. Further, we have that $d_2 \leq v_2$ and $d_2 \leq u_2$.

By repeating the previous arguments, we can show that each element of $S+t^*$, from smallest to largest, must also be an element of $S+v+t^*$.

Thus, $S + v + t^* = S + t^*$ and so $S + v = S$. In (5.13), the union operations are both disjoint, hence $S + v = S$ implies $S + u = S + 1$. Now,

$$\begin{aligned} & S + v = S \\ \implies \quad & 2^v B \equiv B \pmod{2^\ell - 1} \\ \implies \quad & \overset{\hookrightarrow}{\mathrm{ord}}(B) \mid v \\ \implies \quad & \ell \mid v \,. \end{aligned}$$

And similarly, $\ell \mid (u-1)$. Thus $\gcd(u-1, v) \geq \ell > 1$. This proves the lemma. □

Looking at an example can help us connect the different steps in the proof of Theorem 5.24. Suppose $x = 3 - (2^u + 2^v)$ with $u > v \geq 2$. If $\{0, 1, x\}$ is not a NADS then $\exists \alpha \in \{00, 01, 0, 0x\}^*$ such that $(\alpha)_2 \equiv 0 \pmod{2^{|\alpha|} - 1}$. By the definition of x, it must be that $\alpha \in \{00, 0, 0x\}^*$. We will suppose $\alpha = 0x0x000x0x0x000x$, and so $|\alpha| = 16$. Now,

$$\begin{aligned} (0x0x000x0x0x000x)_2 &\equiv 0 \pmod{2^{16} - 1} \\ \implies x(01010001 \| 01010001)_2 &\equiv 0 \pmod{2^{16} - 1} \\ \implies (2^u + 2^v - 3)(2^8 + 1)(01010001)_2 &\equiv 0 \pmod{2^{16} - 1} \\ \implies (2^u + 2^v - 3)(01010001)_2 &\equiv 0 \pmod{2^8 - 1} \\ \implies (2^u + 2^v) \cdot 81 &\equiv (2^1 + 2^0) \cdot 81 \pmod{2^8 - 1} \,. \end{aligned}$$

Note that $(01010001)_2 = 81$ is 8-CNA, and $\overset{\hookrightarrow}{\mathrm{ord}}(81) = 8$. Let $S = \{0, 4, 6\}$, then $d(S) = (4, 2, 2)$ and $d(S + 4) = (2, 2, 4)$ which is the lexicographically smallest cyclic shift of $d(S)$. Continuing from our last implication,

$$\begin{aligned} \implies (S + u) \cup (S + v) &= (S + 1) \cup S \\ \implies (S + u + 4) \cup (S + v + 4) &= (S + 5) \cup (S + 4) \\ \implies (S + u + 4) \cup (S + v + 4) &= \{1, 3, 5\} \cup \{0, 2, 4\} \,. \end{aligned}$$

We can assume that $0 \in S + v + 4$, and then it must be that $1 \in S + u + 4$. If $2 \in S + u + 4$, this would contradict the fact that $(2, 2, 4)$ is the smallest difference sequence of all translates of S. Thus, $2 \in S+v+4$ and then $3 \in S+u+4$. Similarly, $4 \in S+v+4$ and $5 \in S+u+4$. Thus,

$$\begin{aligned} S+u+4 = S+5 \quad &\text{and} \quad S+v+4 = S+4 \\ \implies u \equiv 1 \pmod 8 \quad &\text{and} \quad v \equiv 0 \pmod 8 . \end{aligned}$$

Now, $-x = 2^u + 2^v - 3 = 2(2^{u-1} - 1) + (2^v - 1)$. Since $2^8 - 1 | 2^{u-1} - 1$ and $2^8 - 1 | 2^v - 1$, we have $2^8 - 1 | x$. So, if $\{0, 1, x\}$ is not a NADS, then it must be that x is divisible by a Mersenne number.

If we take $u, v \in \{2, 3, 4, 5, 6, 7, 8\}$ with $u \neq v$ and set $x = 3 - (2^u + 2^u)$ then, after eliminating multiples of Mersenne numbers, Theorem 5.24 tells us that each of the values $-17, -37, -65, -157, -257, -269, -317$ makes $\{0, 1, x\}$ a NADS.

It may not be immediately clear that there are in fact an infinite number of x values with no Mersenne divisors and $w((3-x)/4) = 2$, however, we can deduce this from Lemma 5.26. If we let $-x = 2^u + 2^v - 3$ with $u > v \geq 2$, then, applying Lemma 5.26 with $B = 1$, we see that x has a Mersenne divisor if and only if $\gcd(u, v-1) > 1$ or $\gcd(u-1, v) > 1$. So, we can get an infinite family if we take u's and v's with $\gcd(u, v-1) = \gcd(u-1, v) = 1$. For example, take $v = 2$ and let $u = 4, 6, 8, 10, 12, \ldots$

Looking at Theorems 5.23 and 5.24, a natural question to ask is if there is an infinite family of NADS with the property that $w(\frac{3-x}{4}) = 3$. One of our results gives a partial answer to this question. If $\frac{3-x}{4} = 11 \cdot 2^i$ with $i \geq 0$, then $w(\frac{3-x}{4}) = 3$, however Corollary 5.19 tells us that such a value of x will never give a NADS.

5.8 Further Work and Comments

It is possible to show that for $n \in \mathbb{Z}^+$ with $n \leq \lfloor -x/3 \rfloor$, the running time of EVAL-$R_D(n)$, as described in Algorithm 5.11, is $O(|x|/3) = O(|x|)$. Thus, to compute EVAL-$R_D(n)$ for all positive integers in this range takes time $O(|x|^2)$. So, we can decide if $\{0, 1, x\}$ is a NADS in $O(|x|^2)$ time. The running time can be reduced to $O(|x|)$ if more memory is used, and this is the approach taken in Algorithm 5.14. However, since the size of the input to this algorithm is $\lg |x|$, the running time is exponential. It would be interesting to determine if there is a polynomial-time algorithm for deciding if $\{0, 1, x\}$ is a NADS.

Avoine, Monnerat and Peyrin [3] have significantly improved the search bound presented in Theorem 5.12 and Corollary 5.13:

Theorem 5.28 (Avoine, Monnerat and Peyrin). *Suppose x is a negative integer not divisible by 3 or 7 and $x \equiv 3 \pmod 4$. If all positive integers congruent to 3 modulo 4 that satisfy $n \leq \lfloor -x/12 \rfloor$ or $\lfloor -x/7 \rfloor \leq n \leq \lfloor -x/6 \rfloor$ have $\{0, 1, x\}$-NAF's, then $\{0, 1, x\}$ is a NADS.*

They also conjecture that checking integers in the range $\lfloor -x/7 \rfloor \leq n \leq \lfloor -x/6 \rfloor$ is unnecessary; that is, they conjecture that if $\{0, 1, x\}$ is not a NADS, and x is not divisible by 3 or 7, then there is some integer n congruent to 3 modulo 4 with $n \leq \lfloor -x/12 \rfloor$ that certifies this.

Heuberger and Prodinger [15] have considered the question of which values of x provide each integer with a *minimal weight* $\{0, 1, x\}$-NAF. They show that the only values of x with this property are -1 and 3.

The "NADS-counting" function is defined as follows

$$c(X) := |\{x : x \geq X, \{0, 1, x\} \text{ is a NADS}\}| .$$

For example, $c(7) = 0$, $c(3) = 1$ and $c(-1) = 2$. A plot of $c(X)$ for $0 \geq X \geq -10^7$ is given in Figure 5.5 and an interesting fractal structure can be observed. The flat intervals of the plot are the result of Theorem 5.21. The two smooth curves bounding $c(X)$ in Figure 5.5 are $(-X)^{0.64}$ and $(-X)^{0.66}$; these functions were discovered empirically. It would be nice to be able to say something concrete about the asymptotic behaviour of $c(X)$.

The function f_D, defined in (5.1), bears some similarity to the Collatz function,

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ is even,} \\ (3n+1)/2 & \text{if } n \text{ is odd.} \end{cases}$$

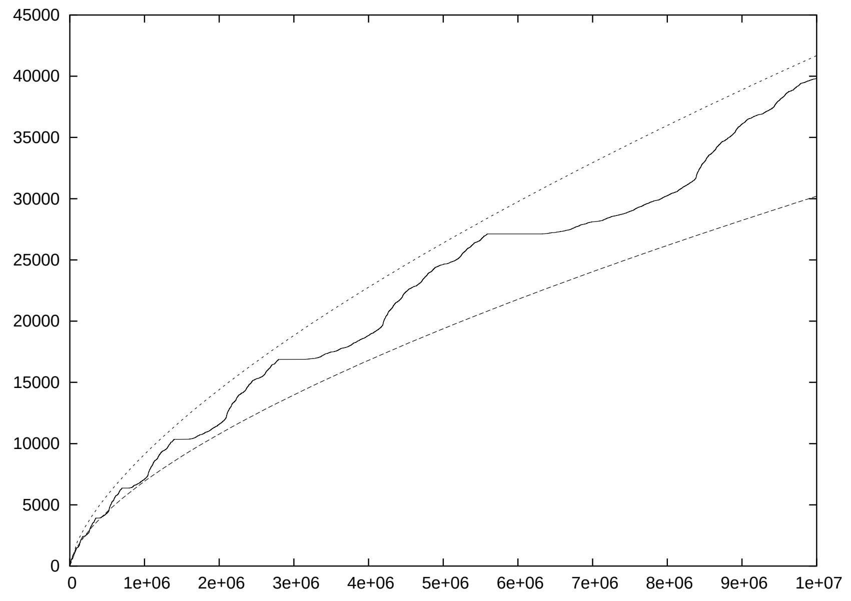

Figure 5.5: A plot of $(-X, c(X))$ for $0 \geq X \geq -10^7$.

The Collatz function has received considerable study, but its properties are complex and not well understood. Perhaps this suggests that the study of NADS is also a difficult problem.

Of the non-allowable factors of x that we discussed, perhaps the more interesting variety of these integers are those for which none of their proper divisors are non-allowable factors. We call a non-allowable factor *simple* if it has this property. It would be interesting to know if there are an infinite number of simple non-allowable factors.

Some of our results on NADS appear to have analogs in Matula's [24] theory on basic digit sets. In particular, our Theorem 5.12 corresponds to Matula's Lemma 6, and our Theorem 5.15 corresponds to Matula's Theorem 5. It would be interesting to find other connections between the two works. It might be that our Theorems 5.23 and 5.24, which do not appear to have analogs in [24], could lead to some new results in the theory of basic digit sets.

Appendix A

The Average Density of the Joint Sparse Form

The *density* of a joint representation $(\mathcal{A})_2$ is defined as

$$\frac{\mathsf{wt}(\mathcal{A})}{\mathsf{length}(\mathcal{A})}.$$

The *average density* of a family of joint representations is often used to calculate the expected running time of algorithms which employ these representations to compute algebraic operations. Solinas [42] uses a probablistic argument to compute the average density of the joint sparse form. Here, we show how this value can be computed using discrete methods.

A.1 The Joint Sparse Form

A representation $(\mathcal{A})_2$, with $\mathcal{A} \in (\{0, \pm 1\}^{2\times 1})^*$, is called a joint sparse form if it satisfies the following three properties:

JS-1 *Of any three consecutive columns, at most two are nonzero.*

JS-2 *No two adjacent digits are* $1\overline{1}$ *or* $\overline{1}1$.

JS-3 *For any two consecutive columns, if one row is* 11 *or* $\overline{11}$, *the other row is* 10 *or* $\overline{1}0$.

To simplify some of our notation, we denote the digits of a JSF as

$$\begin{pmatrix} a_{\ell-1} \dots a_2 a_1 a_0 \\ b_{\ell-1} \dots b_2 b_1 b_0 \end{pmatrix}_2 .$$

A.2 Recurrences

Let r_ℓ denote the number of JSFs of length ℓ, and let w_ℓ denote the number of nonzero columns amongst all JSFs of length ℓ. The average density of an ℓ-column JSF can be computed as $\frac{w_\ell}{\ell \cdot r_\ell}$. To this end, we develop formulae for r_ℓ and w_ℓ.

A.2.1 The Number of JSFs

We start with r_ℓ. To determine r_1 we list all the length-1 joint representations which satisfy JS-1, JS-2 and JS-3. This gives us:

$$r_1 = \left| \left\{ \begin{array}{cccccccc} 0 & 0 & 1 & 1 & 1 & \overline{1} & \overline{1} & \overline{1} \\ 1, & \overline{1}, & 0, & 1, & \overline{1}, & 0, & 1, & \overline{1} \end{array} \right\} \right| = 8.$$

Note that we have omitted the usual parentheses $(\quad)_2$ around the joint representations above.

We determine r_2 in the same way.

$$r_2 = \left| \left\{ \begin{array}{cccccccc} 00 & 00 & 10 & 10 & 10 & \overline{1}0 & \overline{1}0 & \overline{1}0 \\ 10, & \overline{1}0, & 00, & 10, & \overline{1}0, & 00, & 10, & \overline{1}0, \\[1ex] 01 & 0\overline{1} & 01 & 0\overline{1} & 10 & 10 & 10 & 11 \\ 10, & 10, & \overline{1}0, & \overline{1}0, & 01, & 0\overline{1}, & 11, & 10, \\[1ex] 10 & 11 & \overline{1}0 & \overline{1}0 & \overline{1}0 & \overline{11} & \overline{1}0 & \overline{1}0 \\ \overline{11}, & \overline{1}0, & 01, & 0\overline{1}, & 11, & 10, & \overline{1}0, & \overline{11} \end{array} \right\} \right| = 24.$$

To determine r_3, we argue as follows. Let $\genfrac{}{}{0pt}{}{*}{*}$ denote a nonzero column. In a length-3 JSF, $\binom{a_1}{b_1}$ is either a zero or nonzero column. If it is a zero column, then the JSF must look like

$$\genfrac{}{}{0pt}{}{*00}{*00}, \text{ or } \genfrac{}{}{0pt}{}{*0*}{*0*}.$$

Using the fact that $r_1 = 8$, we count 8 and $8 \cdot 8 = 64$ JSFs in each of these two forms.

If $\binom{a_1}{b_1}$ is a nonzero column, condition JS-1 implies that the JSF looks like

$$\begin{matrix} **0 \\ **0 \end{matrix}.$$

To count the number of JSFs of this form, we refer back to the set of all length-2 JSFs. In this set we see there are exactly 16 length-2 JSFs which have both columns nonzero. By appending a zero column to each of these representations, we identify each JSF in this last case. Thus, there are 16 JSFs of this form. Putting everything together, we have

$$\begin{aligned} r_3 &= 8 + 64 + 16 \\ &= 8(r_1 + 1) + 16 = 88. \end{aligned}$$

The same technique can be used to determine r_4. In a length-4 JSF, $\binom{a_2}{b_2}$ is either a zero or nonzero column. If it is a zero column then the JSF must look like

$$\begin{matrix} *000 \\ *000 \end{matrix}, \begin{matrix} *00* \\ *00* \end{matrix}, \text{ or } \begin{matrix} *0*a_0 \\ *0*b_0 \end{matrix}.$$

Of the first and second form, there are 8 and $8 \cdot r_1$ JSFs, respectively. For the third form, each length-2 JSF identifies 8 JSFs of this form. This gives $8 \cdot r_2$ JSFs.

If $\binom{a_2}{b_2}$ is a nonzero column, then the JSF must look like

$$\begin{matrix} **00 \\ **00 \end{matrix}, \text{ or } \begin{matrix} **0* \\ **0* \end{matrix}.$$

There are 16 JSFs of the first type, and $16 \cdot r_1$ JSFs of the second type. Thus, we have

$$\begin{aligned} r_4 &= 8(r_2 + r_1 + 1) + 16(r_1 + 1) \\ &= 408. \end{aligned}$$

In general, for $\ell \geq 4$, we have

$$r_\ell = 8(r_{\ell-2} + r_{\ell-3} + \cdots + r_1 + 1) + 16(r_{\ell-3} + \cdots + r_1 + 1).$$

If $\ell \geq 5$, then

$$r_{\ell-1} = 8(r_{\ell-3} + r_{\ell-4} + \cdots + r_1 + 1) + 16(r_{\ell-4} + \cdots + r_1 + 1),$$

and subtracting these two expressions we find

$$r_\ell - r_{\ell-1} - 8r_{\ell-2} - 16r_{\ell-3} = 0.$$

The characteristic equation of this recurrence is $x^3 - x^2 - 8x - 16 = 0$ which has roots

$$x_1 = 4,\ x_2 = -2e^{-i\theta},\ \text{and}\ x_3 = -2e^{i\theta},\ \text{where}\ \theta = \arctan\left(\frac{\sqrt{7}}{3}\right).$$

We look for a solution of the recurrence of the form $c_1x_1^\ell + c_2x_2^\ell + c_3x_3^\ell$. After some algebraic manipulations, this expression can be rewritten as

$$\hat{c}_1 4^\ell + \hat{c}_2(-2)^\ell \cos \ell\theta + \hat{c}_3(-2)^\ell \sin \ell\theta.$$

We use the values of r_1, r_2 and r_3 to determine the values of the constants. After some calculations, we find

$$r_\ell = \left(\frac{3}{2}\right)4^\ell + \left(\frac{-3}{2}\right)(-2)^\ell \cos \ell\theta + \left(\frac{\sqrt{17}}{14}\right)(-2)^\ell \sin \ell\theta.$$

A.2.2 The Number of Nonzero Columns

Now, we tackle w_ℓ. Referring back to the sets of JSFs defined during the computation of r_1 and r_2, we see that

$$\begin{aligned} w_1 &= 8 \\ w_2 &= 40. \end{aligned}$$

We can determine w_3 as follows. There are r_3 length-3 JSFs and, in each one, $\binom{a_2}{b_2}$ is a nonzero column. Thus,

$$w_3 = r_3 + \text{the number of nonzero columns in positions other than } \binom{a_2}{b_2}.$$

We know the value of r_3. As for the other term, we calculate it in two steps.

First, we consider length-3 JSFs where $\binom{a_1}{b_1}$ a zero column. As noted previously, such length-3 JSFs have the form

$$\begin{matrix} *00 \\ *00 \end{matrix}, \text{ or } \begin{matrix} *0* \\ *0* \end{matrix}.$$

In the first case, neither $\binom{a_0}{b_0}$ or $\binom{a_1}{b_1}$ is ever nonzero. In the second case, $\binom{a_0}{b_0}$ is nonzero $8 \cdot w_1$ times and $\binom{a_1}{b_1}$ is nonzero 0 times.

For the second step, we suppose $\binom{a_1}{b_1}$ is nonzero. Such length-3 JSFs have the form

$$\begin{matrix} **0 \\ **0 \end{matrix}.$$

In this case, we see $\binom{a_0}{b_0}$ is nonzero 0 times, and $\binom{a_1}{b_1}$ is nonzero 16 times. Thus, we have counted every nonzero column occurring in the rightmost two columns. Combining our results, we have

$$w_3 = r_3 + 8 \cdot w_1 + 16 = 168.$$

We use the same approach to determine w_4. As before,

$$w_4 = r_4 + \text{number of nonzero columns in positions other than } \binom{a_3}{b_3}.$$

If $\binom{a_2}{b_2}$ is zero, then such length-4 JSFs have the form

$$\begin{matrix} *000 \\ *000 \end{matrix}, \begin{matrix} *00* \\ *00* \end{matrix}, \text{ or } \begin{matrix} *0*a_0 \\ *0*b_0 \end{matrix}.$$

In the first case, we count 0 nonzero columns other than $\binom{a_3}{b_3}$, in the second case, there are $8 \cdot w_1$, and in the third case, $8 \cdot w_2$.

If $\binom{a_2}{b_2}$ is nonzero, then such length-4 JSFs have the form

$$\begin{matrix} **00 \\ **00 \end{matrix}, \text{ or } \begin{matrix} **0* \\ **0* \end{matrix}.$$

In the first case, $\binom{a_2}{b_2}$ is nonzero 16 times and the other columns, besides $\binom{a_3}{b_3}$, are always zero. In the second case, $\binom{a_0}{b_0}$ is nonzero $16 \cdot w_1$ times and

$\binom{a_1}{b_1}$ is nonzero 0 times. This leaves column $\binom{a_2}{b_2}$. It is nonzero in every representation of this last form. There are $16 \cdot r_1$ such representations, thus the number of times this column is nonzero is $16 \cdot r_1$. Summing, we have

$$\begin{aligned} w_4 &= r_4 + 8 \cdot w_1 + 8 \cdot w_2 + 16 + 16 \cdot w_1 + 16 \cdot r_1 \\ &= r_4 + 8(w_2 + w_1) + 16(w_1 + r_1 + 1) \\ &= 1064. \end{aligned}$$

In general, for $\ell \geq 4$, we have

$$\begin{aligned} w_\ell =& r_\ell + 8(w_{\ell-2} + \cdots + w_1) + \\ & 16(w_{\ell-3} + r_{\ell-3} + w_{\ell-4} + r_{\ell-4} + \cdots + w_1 + r_1 + 1). \end{aligned}$$

If $\ell \geq 5$, then

$$\begin{aligned} w_{\ell-1} =& r_{\ell-1} + 8(w_{\ell-3} + \cdots + w_1) + \\ & 16(w_{\ell-4} + r_{\ell-4} + w_{\ell-5} + r_{\ell-5} + \cdots + w_1 + r_1 + 1). \end{aligned}$$

and substracting these two expressions we find

$$w_\ell - w_{\ell-1} - 8w_{\ell-2} - 16w_{\ell-3} = r_\ell - r_{\ell-1} + 16r_{\ell-3}.$$

This results in a nonhomogenus recurrence relation. The corresponding homogenus recurrence is

$$w_\ell - w_{\ell-1} - 8w_{\ell-2} - 16w_{\ell-3} = 0,$$

which is identical to the recurrence we solved earlier. Recall, the roots of its characteristic equation are

$$x_1 = 4,\ x_2 = -2e^{-i\theta}, \text{ and } x_3 = -2e^{i\theta}, \text{ where } \theta = \arctan\left(\frac{\sqrt{7}}{3}\right).$$

The homogenus recurrence has a solution of the form $c_1x_1^\ell + c_2x_2^\ell + c_3x_3^\ell$. However, the nonhomogenus part of the recurrence, $r_\ell - r_{\ell-1} + 16r_{\ell-3}$, can also be expressed in this form. Thus, to solve the original recurrence we look for a solution of the form

$$c_1x_1^\ell + c_2x_2^\ell + c_3x_3^\ell + c_4\ell x_1^\ell + c_5\ell x_2^\ell + c_6\ell x_3^\ell.$$

After some algebraic manipulations, we can rewrite the form of the solution as

$$\begin{aligned}&\hat{c}_1 4^\ell + \hat{c}_2(-2)^\ell \cos \ell\theta + \hat{c}_3(-2)^\ell \sin \ell\theta + \\ &\hat{c}_4 \ell 4^\ell + \hat{c}_5\ell(-2)^\ell \cos \ell\theta + \hat{c}_6\ell(-2)^\ell \sin \ell\theta.\end{aligned}$$

We can use the recurrence formula to compute the following values: $w_1 = 8, w_2 = 40, w_3 = 168, w_4 = 1064, w_5 = 4520$ and $w_6 = 21800$. These six values can be used to determine the value of the constants above. This involves solving a system of linear equations. After doing so, we find

$$\begin{aligned}w_\ell =&\left(\frac{13}{16}\right)4^\ell + \left(\frac{-13}{16}\right)(-2)^\ell \cos \ell\theta + \left(\frac{169\sqrt{7}}{784}\right)(-2)^\ell \sin \ell\theta + \\ &\left(\frac{3}{4}\right)\ell 4^\ell + \left(\frac{-31}{28}\right)\ell(-2)^\ell \cos \ell\theta + \left(\frac{3\sqrt{7}}{28}\right)\ell(-2)^\ell \sin \ell\theta.\end{aligned}$$

A.3 Average Density

Using our formulae for r_ℓ and w_ℓ, we can now compute the average density of a length-ℓ JSF as follows. We have

$$\frac{w_\ell}{\ell \cdot r_\ell} = \frac{784\, w_\ell}{784\, \ell\, r_\ell}.$$

For the numerator, we have

$$\begin{aligned}784\, w_\ell =&637\, 4^\ell \; - \; 637(-2)^\ell \cos \ell\theta \; + \; 169\sqrt{7}(-2)^\ell \sin \ell\theta + \\ &588\, \ell\, 4^\ell \; - \; 868\, \ell\, (-2)^\ell \cos \ell\theta \; + \; 84\sqrt{7}\, \ell\, (-2)^\ell \sin \ell\theta.\end{aligned}$$

For the denominator, we have

$$784\, \ell\, r_\ell = 1176\, \ell\, 4^\ell \; - \; 1176\, \ell\, (-2)^\ell \cos \ell\theta \; + \; 56\sqrt{7}\, \ell\, (-2)^\ell \sin \ell\theta.$$

When ℓ is large, the numerator is dominated by the term $588\ \ell\ 4^{\ell}$, and the denominator is dominated by the term $1176\ \ell\ 4^{\ell}$. Thus, when ℓ is large we estimate that

$$\frac{w_\ell}{\ell \cdot r_\ell} \approx \frac{588\ \ell\ 4^{\ell}}{1176\ \ell\ 4^{\ell}} = \frac{1}{2}.$$

Appendix B

Some Nonadjacent Digit Sets

Here we list all the values of x from -1 to -10000 for which $\{0, 1, x\}$ is a NADS:

```
   -1    -5   -13   -17   -25   -29   -37   -53   -61   -65
 -113  -121  -125  -137  -145  -149  -157  -233  -241  -253
 -257  -265  -269  -277  -281  -305  -317  -325  -437  -481
 -485  -493  -505  -509  -517  -521  -533  -541  -557  -565
 -601  -605  -613  -629  -641  -653  -673  -821  -869  -913
 -937  -977  -989 -1013 -1021 -1025 -1033 -1037 -1045 -1061
-1073 -1081 -1097 -1117 -1133 -1145 -1165 -1265 -1273 -1277
-1289 -1297 -1325 -1345 -1349 -1357 -1621 -1637 -1733 -1745
-1765 -1885 -1933 -1949 -1985 -1993 -2017 -2021 -2033 -2041
-2045 -2053 -2069 -2101 -2105 -2113 -2129 -2137 -2141 -2153
-2161 -2165 -2173 -2185 -2189 -2197 -2237 -2273 -2285 -2293
-2297 -2321 -2353 -2365 -2369 -2381 -2393 -2405 -2425 -2497
-2525 -2533 -2557 -2593 -2609 -2621 -2641 -2645 -2669 -2677
-2693 -3245 -3265 -3337 -3385 -3421 -3509 -3541 -3557 -3629
-3653 -3673 -3761 -3797 -3853 -3877 -3881 -3917 -3925 -3929
-3961 -4001 -4033 -4037 -4085 -4093 -4097 -4105 -4117 -4121
-4133 -4141 -4145 -4153 -4157 -4201 -4205 -4217 -4253 -4261
-4273 -4285 -4297 -4337 -4345 -4349 -4373 -4393 -4397 -4469
```

```
-4537 -4541 -4573 -4589 -4597 -4601 -4621 -4633 -4645 -4649
-4661 -4693 -4777 -4801 -5021 -5077 -5093 -5101 -5105 -5113
-5129 -5137 -5153 -5165 -5189 -5197 -5213 -5273 -5281 -5365
-5377 -5381 -5393 -5405 -5437 -5441 -6565 -6613 -6773 -6805
-6929 -6973 -7033 -7277 -7333 -7345 -7381 -7393 -7397 -7465
-7477 -7561 -7597 -7613 -7621 -7649 -7741 -7817 -7865 -7877
-7901 -7949 -8045 -8053 -8065 -8069 -8081 -8093 -8101 -8117
-8129 -8165 -8173 -8177 -8185 -8189 -8201 -8213 -8221 -8233
-8237 -8297 -8305 -8317 -8333 -8341 -8369 -8417 -8429 -8437
-8441 -8453 -8485 -8497 -8501 -8573 -8581 -8593 -8597 -8665
-8669 -8681 -8693 -8717 -8725 -8741 -8753 -8789 -8797 -8825
-8837 -8921 -8977 -9089 -9101 -9133 -9157 -9161 -9181 -9209
-9221 -9245 -9341 -9353 -9421 -9425 -9433 -9461 -9473 -9497
-9505 -9509 -9581 -9665 -9673 -9677 -9697 -9761 -9925 -9997
```

Bibliography

[1] R. M. AVANZI. A Note on the Signed Sliding Window Integer Recoding and its Left-to-Right Analogue, in "Selected Areas in Cryptography 2004". To appear in *Lecture Notes in Computer Science*.

[2] R. M. AVANZI. On Multi-Exponentiation in Cryptography, Cryptology ePrint Archive, Report 2002/154. Available from `http://eprint.iacr.org/2002/154`.

[3] G. AVOINE, J. MONNERAT, T. PEYRIN. Advances in Alternative Non-Adjacent Form Representations, in "Progress in Cryptology - INDOCRYPT 2004". To appear in *Lecture Notes in Computer Science*.

[4] D. J. BERNSTEIN. Pippenger's Exponentiation Algorithm, Preprint. Available from `http://cr.yp.to/papers.html`.

[5] I. F. BLAKE, G. SEROUSSI AND N. P. SMART. *Elliptic Curves in Cryptography*, Cambridge University Press, 1999.

[6] A. D. BOOTH. A Signed Binary Multiplication Technique, *Quarterly Journal of Mechanics and Applied Mathematics* **4** (1951), 236–240.

[7] S. H. CHANG AND N. TSAO-WU. Distance and Structure of Cyclic Arithmetic Codes, in "Proceedings of the Hawaii International Conference on System Sciences" (1968), 463–466.

[8] H. COHEN. Analysis of the Flexible Window Powering Algorithm. To appear in *Journal of Cryptology*. Available from `http://www.math.u-bordeaux.fr/~cohen/window.dvi`.

[9] H. COHEN, A. MIYAJI AND T. ONO. Efficient Elliptic Curve Exponentiation Using Mixed Coordinates, in "Advances in Cryptology – ASIACRYPT '98", *Lecture Notes in Computer Science* **1514** (1998), 51–65.

[10] FIPS 186-2. *Digital Signature Standard (DSS)*. Federal Information Processing Standards Publication 186-2, U.S. Department Of Commerce / National Institute of Standards and Technology, 2000. Available from `http://www.csrc.nist.gov/publications/fips/`

[11] D. M. GORDON. A Survey of Fast Exponentiation Methods, *Journal of Algorithms* **27** (1998), 129–146.

[12] P. GRABNER, C. HEUBERGER AND H. PRODINGER Distribution Results for Low-Weight Binary Representations for Pairs of Integers, *Theoretical Computer Science* **319** (2004), 307–331.

[13] P. GRABNER, C. HEUBERGER, H. PRODINGER AND J. THUSWALDNER. Analysis of Linear Combination Algorithms in Cryptography. Submitted. Available from `http://www.opt.math.tu-graz.ac.at/~cheub/publications/`.

[14] D. HANKERSON, A. MENEZES AND S. VANSTONE. *Guide to Elliptic Curve Cryptography*, Springer, 2004.

[15] C. HEUBERGER AND H. PRODINGER. Analysis of Alternative Digit Sets for Nonadjacent Representations. Submitted. Available from `http://www.opt.math.tu-graz.ac.at/~cheub/publications/`.

[16] C. HEUBERGER, R. KATTI, H. PRODINGER, AND X. RUAN. The Alternating Greedy Expansion and Applications to Left-To-Right Algorithms in Cryptography. Submitted. Available from `http://www.opt.math.tu-graz.ac.at/~cheub/publications/`.

[17] J. JEDWAB AND C. J. MITCHELL. Minimum Weight Modified Signed-Digit Representations and Fast Exponentiation, *Electronic Letters* **25** (1989), 1171–1172.

[18] M. JOYE AND S. YEN. Optimal Left-to-Right Binary Signed-Digit Recoding, *IEEE Transactions on Computers* **49** (2000), 740–748.

[19] R. KATTI. Speeding up Elliptic Cryptosystems Using a New Signed Binary Representation for Integers, in "Proceedings of the Euromicro Symposium on Digital System Design – DSD '02", IEEE (2002), 380–384.

[20] R. KATTI, X. RUAN. Left-to-right Binary Signed-Digit Recoding for Elliptic Curve Cryptography, in "Proceedings of the International Symposium on Circuits and Systems – ISCAS '04", IEEE (2004), 365–368.

[21] D. E. KNUTH, *Seminumerical Algorithms*, Volume 2 of *The Art of Computer Programming*, Addison-Wesley, Third edition, 1997.

[22] K. KOYAMA AND Y. TSURUOKA. Speeding up Elliptic Cryptosystems by Using a Signed Binary Window Method, in "Advances in Cryptology – CRYPTO '92", *Lecture Notes in Computer Science* **740** (1993), 345–357.

[23] J. H. VAN LINT, *Introduction to Coding Theory*, 3rd edition, Springer, 1999.

[24] D. W. MATULA. Basic Digit Sets for Radix Representation, *Journal of the Association for Computing Machinery*, **29** (1982), 1131–1143.

[25] A. J. MENEZES, P. C. VAN OORSCHOT, AND S. A. VANSTONE. *Handbook of Applied Cryptography*, CRC Press, 1996.

[26] A. MIYAJI, T. ONO AND H. COHEN Efficient Elliptic Curve Exponentiation, in "Information and Communication Security – ICICS '97", *Lecture Notes in Computer Science* **1334** (1997), 282–290.

[27] B. MÖLLER. Fractional Windows Revisited: Improved Signed-Digit Representations for Efficient Exponentiation, in "Information Security and Cryptology – ICISC 2004". To appear in *Lecture Notes in Computer Science*.

[28] B. MÖLLER. Improved Techniques for Fast Exponentiation, in "Information Security and Cryptology – ICISC 2002", *Lecture Notes in Computer Science* **2587** (2002), 298–312.

[29] F. MORAIN AND J. OLIVOS. Speeding up the Computations on an Elliptic Curve using Addition-Subtraction Chains, *RAIRO Theoretical Informatics and Applications* **24** (1990), 531–543.

[30] J. A. MUIR AND D. R. STINSON. Alternative Digit Sets for Nonadjacent Representations, in "Selected Areas in Cryptography 2003", *Lecture Notes in Computer Science* **3006** (2004), 306–319.

[31] J. A. MUIR AND D. R. STINSON. Alternative Digit Sets for Nonadjacent Representations (extended version). To appear in *SIAM Journal on Discrete Mathematics*. Available from `http://www.uwaterloo.ca/~jamuir/papers.htm`.

[32] J. A. MUIR AND D. R. STINSON. New Minimal Weight Representations for Left-to-Right Window Methods, in "Topics in Cryptology - CT-RSA 2005". To appear in *Lecture Notes in Computer Science*. Available from `http://www.uwaterloo.ca/~jamuir/papers.htm`.

[33] J. A. MUIR AND D. R. STINSON. Minimality and Other Properties of the Width-w Nonadjacent Form. To appear in *Mathematics of Computation*. Available from `http://www.uwaterloo.ca/~jamuir/papers.htm`.

[34] V. MÜLLER. Fast Multiplication on Elliptic Curves over Small Fields of Characteristic Two. *Journal of Cryptology* **11** (1998), 219–234.

[35] C. H. LIM AND P. J. LEE. More Flexible Exponentiation with Precomputation, in "Advances in Cryptology – CRYPTO '94", *Lecture Notes in Computer Science* **839** (1994), 95–117.

[36] K. OKEYA, K. SCHMIDT-SAMOA, C. SPAHN AND T. TAKAGI. Signed Binary Representations Revisited, in "Advances in Cryptology – CRYPTO 2004", *Lecture Notes in Computer Science* **3152** (2004), 123–139.

[37] N. PIPPENGER. On the Evaluation of Powers and Monomials, *SIAM Journal on Computing* **9** (1980), 230–250.

[38] J. A. PROOS. Joint Sparse Forms and Generating Zero Columns when Combing. Technical Report CORR 2003-23 Centre for Applied Cryptographic Research. Available from `http://www.cacr.math.uwaterloo.ca/techreports/2003`.

[39] G. W. REITWIESNER. Binary Arithmetic, in *Advances in Computers, Vol. 1*, Academic Press, 1960, pp. 231–308.

[40] J. A. SOLINAS. An Improved Algorithm for Arithmetic on a Family of Elliptic Curves. In "Advances in Cryptology – CRYPTO '97", *Lecture Notes in Computer Science* **1294** (1997), 357–371. An extended version of the paper is available from `http://www.cacr.math.uwaterloo.ca/techreports/1999`.

[41] J. A. SOLINAS. Efficient Arithmetic on Koblitz Curves. *Designs, Codes and Cryptography* **19** (2000), 195–249.

[42] J. A. SOLINAS. Low-Weight Binary Representations for Pairs of Integers. Technical Report CORR 2001-41 Centre for Applied Cryptographic Research. Available from `http://www.cacr.math.uwaterloo.ca/techreports/2001`.

[43] E. G. STRAUS. Addition Chains of Vectors (problem 5125). *American Mathematical Monthly* **71** (1964), 806–808.