Journal de Théorie des Nombres
de Bordeaux **13** (2001), 27–41

Signed bits and fast exponentiation

par Wieb BOSMA

Résumé. Nous donnons une analyse précise du gain obtenu en utilisant la représentation des entiers sous la forme non-adjacente, plutôt que la représentation binaire, lorsqu'il s'agit de calculer les puissances d'éléments dans un groupe dans lequel l'inversion est facile. En comptant le nombre de multiplications pour un exposant aléatoire ayant un nombre donné de bits dans son écriture binaire, nous obtenons une version précise du résultat asymptotique connu, selon lequel en moyenne, un parmi trois bits signés de la forme non-adjacente n'est pas nul. Cela montre que l'utilisation des bits signés réduit le coût de l'exponentiation d'un neuvième, par rapport à la méthode ordinaire consistant à des élevations au carré et à des multiplications répétées.

Abstract. An exact analysis is given of the benefits of using the non-adjacent form representation for integers (rather than the binary representation), when computing powers of elements in a group in which inverting is easy. By counting the number of multiplications for a random exponent requiring a given number of bits in its binary representation, we arrive at a precise version of the known asymptotic result that on average one in three signed bits in the non-adjacent form is non-zero. This shows that the use of signed bits (instead of bits for ordinary repeated squaring and multiplication) reduces the cost of exponentiation by one ninth.

1. Introduction

To raise elements of a monoid into the power $e > 1$, the method of repeated squaring and multiplication is often employed. To calculate x^e, where $e = \sum_{i=0}^{n} b_i 2^i$, with $b_i \in \{0,1\}$ and $b_n = 1$, the powers

$$y_0 = x^1, y_1 = x^2, y_2 = x^4, \dots, y_n = x^{2^n}$$

are computed by repeated squaring, and x^e is found by taking the product of the y_i for which $b_i = 1$. It is clear that computing x^e this way takes $l(e) - 1$ squarings and $w(e) - 1$ multiplications, where the (binary) *length*

Manuscrit reçu le 22 octobre 1999.
Thanks to Ruud Jeurissen for first versions of proofs and for helpful discussions.

$l(e) = n + 1$ and the *Hamming weight* $w(e)$ are the total number of bits and the number of non-zero bits b_i used to express the exponent e.

If the monoid is a group in which inverses can be computed efficiently, it may be advantageous to use a different representation of the exponent. Writing $e = \sum_{i=0}^{m} s_i 2^i$, where $s_i \in \{-1, 0, 1\}$, we have obtained a *signed bit* representation [2] for e. To determine x^e, again compute

$$y_0 = x^1, y_1 = x^2, y_2 = x^4, \ldots, y_m = x^{2^m}$$

via repeated squaring, and accumulate the product $y_i^{s_i}$ (for the non-zero s_i), which involves an inversion if $s_i = -1$.

The advantage of signed bit representations is that the signed bit weight $w_s(e)$ may be smaller than $w(e)$. Taking $e = 15$ for example, the binary representation consists of four bits equal to 1. But $15 = 2^4 - 1$, so a signed bit representation of weight 2 and length 5 exists. At the cost of one inversion and an extra squaring we have done away with two multiplications.

There exist better ways to compute x^e, using arbitrary addition chains or addition-subtraction chains. We briefly discuss them in Section 3.

A complication in considering signed bits may seem that signed bit representations of integers are by no means unique. Indeed, using that the integer 1 has a representation $1 = 2^k + \sum_{i=0}^{k-1} -1 \cdot 2^i$, for any $k > 1$, it is seen that every integer admits infinitely many signed bit representations. In Section 2 we describe the *non-adjacent form*, which selects a unique signed bit representation for any non-negative integer e. We indicate how it, and a modified version of it, can be determined efficiently, and we show that these special representations have certain optimal properties.

In Sections 4 and 5 we will analyze exactly the weight of non-adjacent forms for integers e. It is shown (in a precise sense) that on average this weight is a third of the length of e, as opposed to a half for the binary form. In general the gain that can be achieved from this in exponentiation will depend on the relative costs of inverting, multiplying, and squaring in the group. The standard application for signed bit exponentiation is to the arithmetic of elliptic curves, [7], [9]. The group of points on an elliptic curve over a field in Weierstrass form has the desired property that inverting is (almost) for free. When inverting is free the results of Section 5 show that a reduction by a ninth in cost, on average, is obtained by using the non-adjacent form rather than the binary form. This makes precise a result that so far only seems to be known heuristically or asymptotically [1], [7], [9]. (Note that in the elliptic curve case squarings are usually slightly more expensive than ordinary multiplications, which means that the cost reduction from using signed bits is in fact less.)

2. Signed Bits

To fix the notation, let a *signed-bit representation of length* $l(e)$ for a positive integer e be a sequence $s_{l(e)-1}, s_{l(e)-2}, \ldots, s_0$ such that $e = \sum_{i=0}^{l(e)-1} s_i 2^i$, with $s_i \in \{-1, 0, 1\}$ and $s_{l(e)-1} = 1$. Sometimes we will write $m = l(e) - 1$; the sequence of signed bits s_i is usually written without comma's with most-significant digit $s_{l(e)-1}$ first. In a sequence of signed bits the symbol $\overline{1}$ will denote -1. Thus $1000\overline{1}$ is a signed bit representation for 15.

As we have seen already, e will in general have signed-bit representations of various lengths; indeed, since we may replace the leading 2^m by $2^{m+1} - 2^m$, a process which can be repeated, we find infinitely many representations for any e, of arbitrary (large enough) length. With our application of minimizing costs of exponentiation in mind, we are particularly interested in *short* representations of *low weight*.

We will call a signed bit representation for e *optimal* if it has least possible weight and among all representations of minimal weight it has minimal length — clearly the length of the binary expansion is a lower bound for the length of a signed-bit representation. But note that optimality does not determine a unique representation in general, as the example $11 = 2^3 + 2 + 1 = 2^3 + 2^2 - 1$ shows.

Let us first worry about uniqueness. The *non-adjacent form* representation is the signed bit representation for e characterized by the property:

$$s_i \neq 0 \quad \Rightarrow \quad s_{i-1} = 0, \quad \text{for } i \geq 1.$$

Proposition 1. *Positive integers have unique non-adjacent form representations.*

Proof. Suppose that there exist positive integers e with two different non-adjacent forms. Among all such e select e_0 having a non-adjacent form of minimal length. The minimality condition requires that the least significant bit in the minimal representation of e_0 differs from that in any other. The only admissible pairs for the two least-significant bits in non-adjacent forms are 00, 01, $0\overline{1}$, 10, $\overline{1}0$; only $\overline{1}0$ and 10 determine the same value modulo 4, but their least-significant bits are equal.

This ends the proof. □

It is easy to obtain the non-adjacent form from the ordinary binary expansion: apply the following rule repeatedly, working from right to left (least-significant first):

$$\textit{replace any sequence } 01\cdots 1 \textit{ by } 10\cdots 0\overline{1}$$

where the number of consecutive 0's in the latter is one less than the number of consecutive 1's in the former.

Since $\sum_{i=0}^{k} = 2^{k+1} - 1$, it is clear that the result will always be a non-adjacent form representation for the given integer determined by the binary expansion. It will also be clear that the length of the non-adjacent form is either equal to or one larger than that of the binary expansion.

Example. Starting with the binary expansion for $3190 = 2^{11} + 2^{10} + 2^6 + 2^5 + 2^4 + 2^2 + 2$, the rule produces:

$$\begin{array}{ccccccccccccc}
 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\
 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & \overline{1} & 0 \\
 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & \overline{1} & 0 & \overline{1} & 0 \\
1 & 0 & \overline{1} & 0 & 0 & 1 & 0 & 0 & 0 & \overline{1} & 0 & \overline{1} & 0
\end{array}$$

for $3190 = 2^{12} - 2^{10} + 2^7 - 2^3 - 2$.

In fact the above procedure can be generalized to transform any given signed bit representation into the non-adjacent form; first apply the following rule repeatedly working from left to right:

$$\text{(I)} \qquad \begin{array}{ll} \textit{replace } \overline{1}1 & \textit{by } 0\overline{1}, \quad \textit{and} \\ \textit{replace } 1\overline{1} & \textit{by } 01, \end{array}$$

and then apply the following repeatedly (working from right to left):

$$\text{(II)} \qquad \begin{array}{llll} \textit{replace} & 0\overbrace{1\cdots1}^{k>1} & \textit{by} & 1\overbrace{0\cdots0}^{k-1}\overline{1}, \quad \textit{and} \\ \textit{replace} & 0\underbrace{\overline{1}\cdots\overline{1}}_{k>1} & \textit{by} & \overline{1}\underbrace{0\cdots0}_{k-1}1, \end{array}$$

followed by a step of the form (I) if necessary.

Proposition 2. *For any integer the non-adjacent form has minimal weight.*

Proof. Apply the above two rule-transformation to any signed bit representation of minimal weight; the result is the non-adjacent form. The transformation does not increase the weight. □

Corollary 3. *For every integer there is a unique signed bit representation satisfying:*

$$s_k \neq 0 \quad \Rightarrow \quad s_{k-1} = 0, \quad \text{or} \quad k = m \quad \text{and} \quad s_m = s_{m-1} = 1;$$

moreover this expansion is optimal.

Proof. Let $t_m t_{m-1} \cdots t_1 t_0$ be the non-adjacent form for e. If the three most significant bits $t_m t_{m-1} t_{m-2}$ are $10\overline{1}$, then let $n = m - 1$ and define

$$s_i = \begin{cases} 1 & \text{for } i = n, n-1 \\ t_i & \text{for } 0 \leq i \leq n-2. \end{cases}$$

In all other cases let $n = m$ and $s_i = t_i$ for $0 \leq i \leq n$. This way s is equal to the non-adjacent form except when the leading digits for the non-adjacent form are $10\bar{1}0$, in which case we replace them by the shorter expansion with leading digits 110. Clearly s satisfies the non-adjacency conditions of the statement; we will show that it is optimal too.

In the exceptional case the weights of s and t are equal, but the length of s equals that of the binary expansion. Hence s is optimal in that case. We will prove that in all other cases the non-adjacent form t itself is optimal.

Suppose that e is an integer with non-adjacent form $t_m t_{m-1} \cdots t_1 t_0$ of minimal length that is not optimal. Since the non-adjacent weight *is* always minimal, this can only occur if the length of the non-adjacent form of e exceeds that of its binary expansion by 1. This only happens if in the final transformation step a sequence of $k \geq 2$ adjacent 1's is replaced by $10 \cdots 0\bar{1}$, where the number of 0's is $k-1$. If $k=2$ we are in the exceptional case, so we will assume that $k > 2$. The binary expansion $u_{m-1}u_{m-2} \cdots u_0$ has $u_{m-1} = u_{m-2} = u_{m-3} = 1$, while $u_{m-4} = 0$ or 1.

Since the non-adjacent weight *is* minimal, there must exist a signed bit representation $v_{m-1}v_{m-2} \cdots v_0$ of length m, and it necessarily has $v_{m-1} = v_{m-2} = v_{m-3} = 1$, and $v_{m-4} = u_{m-4} \in \{0, 1\}$ since u and v represent the same number e. If $v_{m-4} = 1$, an extra reduction step reduces length plus weight, which contradicts optimality of v. So $v_{m-4} = 0$; but then $v \neq u$ contradicts minimality of m since $v_{m-5}v_{m-6} \cdots$ represents the same number as $u_{m-5}u_{m-6}$ with lower weight.

That ends the proof. □

We will refer to the optimal representation of Corollary 3 as the *modified non-adjacent form*. It is the same as the non-adjacent form, except that non-adjacency is allowed in the most significant two bits, that is 110 is not transformed to $10\bar{1}0$, because such transformation increases the length without decreasing the weight.

Note that this does *not* mean that the modified version is different for precisely those integers for which the leading bits in the binary expansion are 110 because of the propagation of carries in the transformations: non-adjacent and modified non-adjacent forms for $27 = 11011 = 100\bar{1}0\bar{1}$ are the same, but for $25 = 11001$ they are different, namely $10\bar{1}001$ and 11001.

It is not so difficult to obtain the (modified) non-adjacent form directly from e, without computing the binary (or another signed-bit) expansion first. The method resembles the method for finding the binary expansion producing the least significant bit first: starting with $k = e$ repeat:

if k even: produce 0 and divide k by 2;
if k odd: produce 1, subtract 1 from k and divide k by 2;

until k is 0.

For the non-adjacent form one proceeds as follows. Starting with $k = e > 0$ again, one repeats:

e	binary	NAF	modified NAF
1	1	1	1
2	1 0	1 0	1 0
3	1 1	1 0 $\bar{1}$	1 1
4	1 0 0	1 0 0	1 0 0
5	1 0 1	1 0 1	1 0 1
6	1 1 0	1 0 $\bar{1}$ 0	1 1 0
7	1 1 1	1 0 0 $\bar{1}$	1 0 0 $\bar{1}$
8	1 0 0 0	1 0 0 0	1 0 0 0
9	1 0 0 1	1 0 0 1	1 0 0 1
10	1 0 1 0	1 0 1 0	1 0 1 0
11	1 0 1 1	1 0 $\bar{1}$ 0 $\bar{1}$	1 1 0 $\bar{1}$
12	1 1 0 0	1 0 $\bar{1}$ 0 0	1 1 0 0
13	1 1 0 1	1 0 $\bar{1}$ 0 1	1 1 0 1
14	1 1 1 0	1 0 0 $\bar{1}$ 0	1 0 0 $\bar{1}$ 0
15	1 1 1 1	1 0 0 0 $\bar{1}$	1 0 0 0 $\bar{1}$
16	1 0 0 0 0	1 0 0 0 0	1 0 0 0 0
17	1 0 0 0 1	1 0 0 0 1	1 0 0 0 1
18	1 0 0 1 0	1 0 0 1 0	1 0 0 1 0
19	1 0 0 1 1	1 0 1 0 $\bar{1}$	1 0 1 0 $\bar{1}$
20	1 0 1 0 0	1 0 1 0 0	1 0 1 0 0
21	1 0 1 0 1	1 0 1 0 1	1 0 1 0 1
22	1 0 1 1 0	1 0 $\bar{1}$ 0 $\bar{1}$ 0	1 1 0 $\bar{1}$ 0
23	1 0 1 1 1	1 0 $\bar{1}$ 0 0 $\bar{1}$	1 1 0 0 $\bar{1}$
24	1 1 0 0 0	1 0 $\bar{1}$ 0 0 0	1 1 0 0 0
25	1 1 0 0 1	1 0 $\bar{1}$ 0 0 1	1 1 0 0 1
26	1 1 0 1 0	1 0 $\bar{1}$ 0 1 0	1 1 0 1 0
27	1 1 0 1 1	1 0 0 $\bar{1}$ 0 $\bar{1}$	1 0 0 $\bar{1}$ 0 $\bar{1}$
28	1 1 1 0 0	1 0 0 $\bar{1}$ 0 0	1 0 0 $\bar{1}$ 0 0
29	1 1 1 0 1	1 0 0 $\bar{1}$ 0 1	1 0 0 $\bar{1}$ 0 1
30	1 1 1 1 0	1 0 0 0 $\bar{1}$ 0	1 0 0 0 $\bar{1}$ 0
31	1 1 1 1 1	1 0 0 0 0 $\bar{1}$	1 0 0 0 0 $\bar{1}$
32	1 0 0 0 0 0	1 0 0 0 0 0	1 0 0 0 0 0
33	1 0 0 0 0 1	1 0 0 0 0 1	1 0 0 0 0 1
34	1 0 0 0 1 0	1 0 0 0 1 0	1 0 0 0 1 0
35	1 0 0 0 1 1	1 0 0 1 0 $\bar{1}$	1 0 0 1 0 $\bar{1}$
36	1 0 0 1 0 0	1 0 0 1 0 0	1 0 0 1 0 0
37	1 0 0 1 0 1	1 0 0 1 0 1	1 0 0 1 0 1
38	1 0 0 1 1 0	1 0 1 0 $\bar{1}$ 0	1 0 1 0 $\bar{1}$ 0
39	1 0 0 1 1 1	1 0 1 0 0 $\bar{1}$	1 0 1 0 0 $\bar{1}$

$k \bmod 4 \equiv s \in \{-1, 1\}$*:* produce signed bits s and 0, and replace k by $(k-s)/4$;

$k \bmod 4 \equiv s \in \{0, 2\}$*:* produce 0 and replace k by $k/2$.

until k is less than or equal to 3, after which

if $k = 0$*:* produce nothing;
if $k = 1$*:* produce 1;
if $k = 2$*:* produce 0 and 1;
if $k = 3$*:* produce $\overline{1}$ and 0 and 1;

and terminate.
For the modified version the only change necessary is to produce 11 in the case that $k = 3$.

Note the similarities with the continued fraction algorithm, where division by 2 is replaced by inverting, and truncation replaces extracting bits. The algorithm to obtain the non-adjacent form is similar to the nearest integer continued fraction algorithm.

The table shows binary expansion, non-adjacent form, and modified non-adjacent form for the first few positive integers.

3. Addition-subtraction chains

The method of repeated squaring and multiplication does not necessarily give the fastest way to evaluate powers. It is well-known [6] that there are ways to find x^e using fewer multiplications.

An *addition chain* for a positive integer e is a sequence $1 = e_0, e_1, \ldots, e_k = e$ with the property that for $1 \le i \le k$ it holds that $e_i = e_u + e_v$ with $0 \le u, v < i$. Each term is thus the sum of two (possibly the same) previous terms. One usually arranges the e_i in ascending order. The length of the addition chain is the integer k. It will be clear that an addition chain for e can be used to compute x^e: for any i the power x^{e_i} can be computed from $x^{e_0}, \ldots, x^{e_{i-1}}$ by a single multiplication.

The binary expansion $e = \sum_{i=0}^{n} b_i 2^i$ of any e of length $n+1$ defines an addition chain of length $n + w(e) - 1$ for e, corresponding to repeated squaring and multiplication as described in Section 1, as follows. Write down the powers $p_i = 2^i, i = 0, \ldots, n$ of 2 less than or equal to e. Next take $r_0 = 0$ and let r_j be $r_{j-1} + p_{i_j}$, where $i_1, \ldots, i_k$ are those i from 0 to n for which $b_i \neq 0$. The addition chain for e then consists of the the p_i (with $1 \le i \le n$) and r_j (with $j \ge 1$) in ascending order.

There is an alternative addition chain associated with the binary expansion, obtained by reading the bits from left to right (most significant first). Starting with $e_0 = 1$ one repeats for $i = 1, \ldots, n$:

if $b_{n-i} = 1$*:* append $2e_j$ and $2e_j + 1$ to the existing sequence $e_0, \ldots e_j$;
otherwise: append $2e_j$ to the existing sequence $e_0, \ldots e_j$.

There are two problems with general addition chains. In the first place is it hard to find a shortest chain for given e [6]. Secondly, general addition chains make it necessary to remember entries $x^{e_0}, \ldots, x^{e_{i-1}}$ along the way to compute x_i^e. Note that this is not true for the left-to-right binary addition chain, as e_i is either $2e_{i-1}$ or $e_{i-1}+1$, that is, every step is either a squaring or a multiplication by x ([4], see also [8] for the special case of integer exponentiation).

Taking the possibility of subtracting into account as well, we arrive at *addition-subtraction chains* [11]. In general we cannot insist on ascending entries anymore. Again, it will be clear that any signed-bit representation of e will give rise to two addition-subtraction chains, by reading the signed bits either way. It is also obvious that, since the weight of a signed bit representation can be smaller than that of the binary expansion, that the corresponding chain may be shorter.

Examples. Let $e = 43$; reading its bits 101011 right-to-left to obtain the sequence of p_i's $1, 2, 4, 8, 16, 32$ and of r_j's $3, 11, 43$, we obtain an addition chain by merging and ordering: $1, 2, 3, 4, 8, 11, 16, 32, 43$ of length 8.

Reading the binary expansion 101011 left-to-right produces $e_0 = 1$, then $e_1 = 2$, and $e_2 = 4, e_3 = 5$, then $e_4 = 10$, and $e_5 = 20, e_6 = 21$, and finally $e_7 = 42, e_8 = 43$. Indeed, length 8 for 5 doublings and 3 multiplications.

Reading the modified non-adjacent form $43 = 110\bar{1}0\bar{1}$ left-to-right gives the addition-subtraction chain $1, 2, 3, 6, 12, 11, 22, 44, 43$, reading it right-to-left the chain -1, 2, 4, -5, 8, 16, 11, 32, 43. Both have length 8. The non-adjacent form gives chains of length 9.

There exists an addition chain of length 7 for 43: $1, 2, 4, 8, 9, 17, 34, 43$.

The addition-subtraction chain $1, 2, 4, 8, 16, 15$ associated with $15 = 2^4 - 2^0$, is shorter than the chain $1, 2, 3, 6, 7, 14, 15$ arising from the binary expansion $15 = 2^3 + 2^2 + 2^1 + 2^0$. In this case there is an addition chain of length 5 as well, however: $1, 2, 3, 5, 10, 15$ for example.

In general, for $e = 2^k - 1$ the binary expansion gives rise to an addition chain of length $2k - 2$ while the non-adjacent form leads to an addition-subtraction chain of length $k + 1$.

Outside numbers of this form, $e = 23$ is the first example where the modified non-adjacent form for e leads to an addition-subtraction chain ($1, 2, 3, 6, 12, 24, 23$ of length 6) that is strictly shorter than the binary addition chains ($1, 2, 4, 5, 10, 11, 22, 23$ and $1, 2, 3, 4, 7, 8, 16, 23$ of length 7). Again there exist addition chains of length 6, like $1, 2, 3, 5, 10, 13, 23$.

For $e = 27$ there are addition chains (such as $1, 2, 3, 6, 9, 18, 27$) that are shorter than both the chains obtained from the binary expansion ($1, 2, 3, 6, 12, 13, 26, 27$) and the addition-subtraction chain gotten from the (modified) non-adjacent form (1, 2, 4, 8, 7, 14, 28, 27). For $e = 47$ the length of the chain given by the modified non-adjacent form (1, 2, 3, 6, 12, 24, 48, 47) is shorter than any addition chain (the shortest of which have length 8: $1, 2, 3, 4, 7, 10, 20, 27, 47$ for example, while the binary gives length 9: $1, 2, 4, 5, 10, 11, 22, 23, 46, 47$); in this case there is no shorter addition-subtraction chain either.

There are methods to construct short addition chains — these are not necessarily shortest, but shorter than those obtained from the binary expansion. The results from the present paper indicate the gain that can be obtained by using signed bits rather than bits; perhaps this will lead to a more general analysis of the advantages of addition-subtraction chains over addition chains (in situations where they are applicable).

4. Analysis

To analyze the benefits of using the signed bit representations, we first prove some results on (average) length of non-adjacent and modified non-adjacent forms. Let c_n denote the number of positive integers requiring *exactly* n bits in their binary representation, and let c'_n and c''_n be the number of positive integers requiring *exactly* n signed bits in the non-adjacent form and in the modified non-adjacent form representation, respectively. Also, let C_n, C'_n and C''_n similarly define the number of positive integers requiring *at most* n bits in the three representations.

Proposition 4. *The number of positive integers with expansions of length n is given by $c_1 = c'_1 = c''_1 = 1$, and for $n \geq 2$:*

$$c_n = 2^{n-1}, \qquad c'_n = \frac{2}{3}2^{n-1} - \frac{(-1)^n}{3}, \qquad c''_n = \frac{5}{6}2^{n-1} + \frac{(-1)^n}{3}.$$

Hence, for $n \geq 0$:

$$C_n = 2^n, \qquad C'_n = \frac{2}{3}2^n + \frac{1}{2} - \frac{(-1)^n}{6}, \qquad C''_n = \frac{5}{6}2^n + \frac{1}{2} + \frac{(-1)^n}{6}.$$

Proof. Only 1 requires one bit in any expansion. It is also clear that there are exactly 2^{n-1} integers with most significant bit $b_{n-1} = 1$ (of length n), so $c_n = 2^{n-1}$ and $C_n = \sum_{k=0}^{n} c_k = 2^n$.

The easiest way to count integers with n signed bits in their non-adjacent form is to observe that the following recursion holds:

$$c'_{n+2} = c'_{n+1} + 2c'_n, \qquad \text{for} \quad n \geq 1.$$

Namely, the c'_n positive integers of length n (all having $s_{n-1} = 1$), when 'prepended' with $s_n = 0$ and $s_{n+1} = 1$ all contribute. We get another contribution of size c'_n by flipping the n-th bit b_{n-1} to -1. This accounts for all positive integers requiring $n + 2$ bits for which $b_{n-1} \neq 0$. We obtain those with $b_{n-1} = 0$ by taking the c'_{n+1} representations of length $n + 1$ and replacing the leading digit $b_n = 1$ by $b_n = 0$ and putting $b_{n+1} = 1$. This way the validity of the recursion can be seen to hold. With starting values $c'_1 = c'_2 = 1$ the closed form for c'_n in the statement of the proposition is then easily proved, for example by induction. The formula for C'_n is simply obtained by summation: $\sum_{k=0}^{n} c'_k$.

One way to count integers with modified non-adjacent form of length n is to use that their number also satisfies the recursion:

$$c''_{n+2} = c''_{n+1} + 2c''_n, \qquad \text{for} \quad n \geq 2.$$

This time one takes the representations of length n, and obtains from each two valid representations of length $n+2$ by shifting over 2 places and inserting $b_1 = 0$ and $b_0 = \pm 1$. From the length $n+1$ representations one gets length $n+2$ representations by shifting one place and taking $b_0 = 0$. This clearly leads to $2c''_n + c''_{n+1}$ valid representations of length $n+2$ (taking care that $n > 1$ to prevent the illegal representation $10\overline{1}$ for 3) that are all distinct (look at b_0); it is not terribly hard to see that we obtain *all* valid modified signed bit representations this way. The starting values for the recursion are $c''_2 = 2$ and $c''_3 = 3$. Again, C''_n can be derived by summation. $\square$

Here are the first few values for each of the functions:

$$\begin{array}{rl}
n & = 1\ 2\ 3\ \ 4\ \ 5\ \ 6\ \ \ 7\ \ \ 8\ \ \ 9\ \ \ 10\ \ \ 11\ \ldots \\
c_n & = 1\ 2\ 4\ \ 8\ 16\ 32\ \ 64\ 128\ 256\ \ 512\ 1024\ \ldots \\
c'_n & = 1\ 1\ 3\ \ 5\ 11\ 21\ \ 43\ \ 85\ 171\ \ 341\ \ 683\ \ldots \\
c''_n & = 1\ 2\ 3\ \ 7\ 13\ 27\ \ 53\ 107\ 213\ \ 427\ \ 853\ \ldots \\
C_n & = 2\ 4\ 8\ 16\ 32\ 64\ 128\ 256\ 512\ 1024\ \ldots \\
C'_n & = 2\ 3\ 6\ 11\ 22\ 43\ \ 86\ 171\ 342\ \ 683\ \ldots \\
C''_n & = 2\ 4\ 7\ 14\ 27\ 54\ 107\ 214\ 427\ \ 854\ \ldots
\end{array}$$

Remarks. Note that c_n *also* satisfies the recursion that c'_n and c''_n satisfy. The sequence c'_n has been called the Jacobsthal sequence (A001045 in [10]; see also [5]).

Next we count the total weight of all representations of fixed length. Define s_n to be the total number of ones in all different n-bit integers; we use s'_n and s''_n for the total number of non-zero signed bits in all different non-adjacent forms and modified non-adjacent forms of length n. Similarly, by S_n, S'_n and S''_n we denote the total number of non-zeroes in in all binary, non-adjacent and modified non-adjacent representations of length *at most* n.

Proposition 5. *For $n \geq 2$:*

$$\begin{aligned}
s_n &= \frac{n+1}{2} \cdot 2^{n-1}, \\
s'_n &= \frac{6n+10}{27} \cdot 2^{n-1} + (-1)^{n-1} \frac{6n+5}{27}, \\
s''_n &= \frac{15n+34}{54} \cdot 2^{n-1} - (-1)^{n-1} \frac{6n+5}{27}.
\end{aligned}$$

Also,

$$\begin{aligned} S_n &= \frac{n}{2}\cdot 2^n, \\ S'_n &= \frac{6n+4}{27}\cdot 2^n + (-1)^{n-1}\frac{3n+4}{27}, \\ S''_n &= (\frac{5n}{18}+\frac{19}{54})\cdot 2^n - (-1)^{n-1}\frac{3n+4}{27}. \end{aligned}$$

Proof. To count the total number of non-zero bits in n-bit words, note that $n+1$ bit words can be formed out of n-bit words by shifting and 'appending' a single bit (0 or 1). Since there are c_n such n-bit integers, having s_n non-zero bits, we find

$$s_{n+1} = s_n + (s_n + c_n).$$

From $s_1 = 1$ and $s_2 = 3$ we get the result by induction.

To prove the formula for s'_n, note that

$$s'_{n+2} = 2(s'_n + c'_n) + s'_{n+1}.$$

This follows immediately from the proof of the previous Proposition. Then use verification of $s'_1 = s'_2 = 1$ and induction.

For s''_n one derives similarly that

$$s''_{n+2} = s'_n + c'_n + 2\cdot s'_{n+1} + c'_{n+1}.$$

For S_n and S'_n we sum $\sum_{k=0}^{n} s_k$ and $\sum_{k=0}^{n} s'_k$, only using that

$$\sum_{k=0}^{n} k2^k = (n-1)2^{n+1} + 2.$$

$\square$

Here are the first few values for each of the functions again:

n	= 1	2	3	4	5	6	7	8	9	10	11	...
s_n	= 1	3	8	20	48	112	256	576	1280	2816	6144	...
s'_n	= 1	1	5	9	25	53	125	273	609	1325	2885	...
s''_n	= 1	3	5	15	31	75	163	367	799	1747	3771	...
S_n	= 1	4	12	32	80	192	448	1024	2304	5120	...	
S'_n	= 1	2	7	16	41	94	219	492	1101	2426	...	
S''_n	= 1	4	9	24	55	130	293	660	1459	3206	...	

As a consequence we can determine how many non-zero (signed) bits there are on average in all integers requiring exactly or at most n bits in the various expansions; we denote these by g_n, g'_n, g''_n and t_n, t'_n, t''_n.

Corollary 6. *For all $n \geq 2$:*

$$\begin{aligned}
g_n = \frac{s_n}{nc_n} &= \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{n}, \\
g'_n = \frac{s'_n}{nc'_n} &= \frac{1}{3} + \frac{5}{9} \cdot \frac{1}{n} - (-1)^n \frac{1}{3 \cdot (2^n - (-1)^n)}, \\
g''_n = \frac{s''_n}{nc''_n} &= \frac{1}{3} + \frac{34}{45} \cdot \frac{1}{n} + (-1)^n \frac{(1 - \frac{3}{5n})}{3 \cdot (5 \cdot 2^{n-2} + (-1)^n)},
\end{aligned}$$

and

$$\begin{aligned}
G_n = \frac{S_n}{nC_n} &= \frac{1}{2}, \\
G'_n = \frac{S'_n}{nC'_n} &= \frac{1}{3} + \frac{2}{9} \cdot \frac{1}{n} - \frac{3 + (-1)^n + (1 + (-1)^n)\frac{2}{n}}{3 \cdot (2^{n+2} + 3 + (-1)^n)}, \\
G''_n = \frac{S''_n}{nC''_n} &= \frac{1}{3} + \frac{19}{45} \cdot \frac{1}{n} - \frac{3 - (-1)^n + (19 - (-1)^n \cdot 7)\frac{1}{5n}}{3 \cdot (5 \cdot 2^n + 3 + (-1)^n)}
\end{aligned}$$

This Corollary, the proof of which is an easy computation, tells us that on average half the bits in a binary expansion are non-zero (as expected), one in three signed bits in the non-adjacent form are non-zero (compare [1, 3, 9]). For the modified non-adjacent form also a third of the bits are non-zero asymptotically, but the convergence is slightly slower because there are fewer zeroes in the exceptional case.

To give a fair comparison, we need to count the number of bits used for integer with binary expansion of length n. An *n-bit integer* is a non-negative integer for which the ordinary binary representation has length n exactly.

5. Analysis for integers of given length

First we count the total length and the total weight of n-bit integers in the various representations. As usual we denote by l, l', l'' and L, L', L'' the values for ordinary binary, non-adjacent form and modified non-adjacent form representation.

Proposition 7. *The total length of all numbers that take exactly n bits in binary:*

$$\begin{aligned}
l_n &= n2^{n-1}, \\
l'_n &= (n + \frac{2}{3})2^{n-1} - \frac{1}{2} - (-1)^{n-1}\frac{1}{6}, \\
l''_n &= (n + \frac{1}{3})2^{n-1} - \frac{1}{2} + (-1)^{n-1}\frac{1}{6}.
\end{aligned}$$

The total length of all numbers that take at most n bits (in the ordinary representation):

$$\begin{aligned} L_n &= (n-1)2^n + 1, \\ L'_n &= (n-\frac{1}{3})2^n - \frac{n}{2} + \frac{1}{4} + \frac{(-1)^n}{12}, \\ L''_n &= (n-\frac{2}{3})2^n - \frac{n}{2} + \frac{3}{4} - \frac{(-1)^n}{12}. \end{aligned}$$

Proof. Obviously the c_n length n integers give

$$l_n = nc_n.$$

One way to count l'_n is to determine which length n integers contribute to length n non-adjacent forms. These are the binary expansions of length n for which $b_{n-2} = 0$ and for which the non-adjacent form of $b_{n-3}b_{n-4}\cdots b_0$ has length $n-2$. Of those there are exactly C'_{n-2}. The others, $c_n - C'_{n-2} = C_{n-1} - C'_{n-2} = C'_{n-1} - 1$ in number (compare $(*)$), contribute length $n+1$ each, so

$$l'_n = nC'_{n-2} + (n+1)(C'_{n-1} - 1) = (n+1)c_n - C'_{n-2}.$$

Using Proposition 4 immediately gives the desired result.

Similarly it can be proven that

$$l''_n = nC'_{n-1} + (n+1)(C'_{n-2} - 1),$$

For L_n we merely sum:

$$L_n = \sum_{k=0}^{n} l_k,$$

and likewise for L'_n and L''_n. $\square$

The first few values for these functions are:

n	= 1	2	3	4	5	6	7	8	9	10	11	...
l_n	= 1	4	12	32	80	192	448	1024	2304	5120	11264	...
l'_n	= 1	5	14	37	90	213	490	1109	2474	5461	11946	...
l''_n	= 1	4	13	34	85	202	469	1066	2389	5290	11605	...
L_n	= 1	5	17	49	129	321	769	1793	4097	9217	...	
L'_n	= 1	6	20	57	147	360	850	1959	4433	9894	...	
L''_n	= 1	5	18	52	137	339	808	1874	4263	9553	...	

Let w_n, w'_n, w''_n denote the total weight of all non-negative integers requiring exactly n bits in binary representation, and W_n, W'_n, W''_n the same for integers of at most n bits.

Proposition 8.

$$w_n = (n+1)2^{n-2}, \qquad w'_n = w''_n = (\frac{n}{3} + \frac{7}{9})2^{n-1} + (-1)^n \frac{1}{9}$$

$$W_n = n2^{n-1}, \qquad W'_n = W''_n = (\frac{n}{3} + \frac{4}{9})2^n - \frac{1}{2} + (-1)^n \frac{1}{18}$$

Proof. Obviously again,

$$w_n = s_n.$$

The weight of non-adjacent and modified non-adjacent forms are the same, so $w'_n = w''_n$ and $W'_n = W''_n$. The first integer that requires n binary bits is $f_n = 2^{n-1}$. For every integer h larger than f_n for which the length of its non-adjacent form is n, there is an integer g smaller than f_n that has non-adjacent form of length $n-1$ and the same weight as h: simply reverse all bits of h except for the most significant one. Thus the integers with non-adjacent forms of length n other than f_n (which has weight 1) contribute exactly half their total weight, that is $(s'_n - 1)/2$, to w'_n. On the other hand, for the same reason exactly half the total weight of the length $n+1$ non-adjacent forms contribute to the binary length n count, which implies that

$$w'_n = \frac{s'_n - 1}{2} + \frac{s'_{n+1} - 1}{2} + 1,$$

the $+1$ being the contribution of f_n itself. Substitution then gives the result. □

A small table again:

n	= 1	2	3	4	5	6	7	8	9	10	11	...
w_n	= 1	3	8	20	48	112	256	576	1280	2816	6144	...
$w'_n = w''_n$	= 1	3	7	17	39	89	199	441	967	2105	4551	...
W_n	= 1	4	12	32	80	192	448	1024	2304	5120	...	
$W'_n = W''_n$	= 1	4	11	28	67	156	355	796	1763	3868	...	

Corollary 9. *The number of multiplications necessary to compute x^e for a random integer e of exactly n bits using the binary expansion, the non-adjacent form and the modified non-adjacent form for e is:*

$$\begin{aligned} m_n = \frac{l_n + w_n}{c_n} - 2 &= \frac{3}{2}(n-1), \\ m'_n = \frac{l'_n + w'_n}{c_n} - 2 &= \frac{4}{3}(n-1) + \frac{7}{9} - (\frac{1}{2} + (-1)^{n-1}\frac{1}{18}) \cdot \frac{1}{2^{n-1}}, \\ m''_n = \frac{l''_n + w''_n}{c_n} - 2 &= \frac{4}{3}(n-1) + \frac{4}{9} - (\frac{1}{2} - (-1)^{n-1}\frac{5}{18}) \cdot \frac{1}{2^{n-1}}. \end{aligned}$$

If e is random of at most n digits, the cost functions are:

$$\begin{aligned}
M_n = \frac{L_n + W_n}{C_n} - 2 &= \frac{3}{2}(n-2) + \frac{1}{2^n},\\
M'_n = \frac{L'_n + W'_n}{C_n} - 2 &= \frac{4}{3}(n-2) + \frac{7}{9} + (-\frac{n}{2} - \frac{1}{4} + (-1)^n \frac{5}{36}) \cdot \frac{1}{2^n},\\
M''_n = \frac{L''_n + W''_n}{C_n} - 2 &= \frac{4}{3}(n-2) + \frac{4}{9} + (-\frac{n}{2} + \frac{1}{4} - (-1)^n \frac{1}{36}) \cdot \frac{1}{2^n}.
\end{aligned}$$

As expected we see that, for e of binary length n, it takes $n-1$ multiplications (all squarings) and on average $(n-1)/2$ multiplications using the binary expansion for e; using the non-adjacent form the number of multiplications can be reduced to $(n-1)/3$, where on average we save $1/3$ multiplication using the modified form.

References

[1] S. Arno, F. S. Wheeler, *Signed digit representations of minimal Hamming weight.* IEEE Transactions on Computers **42** (1993), 1007–1010.

[2] A. D. Booth, *A signed binary multiplication technique.* Quart. Journ. Mech. and Applied Math. **4** (1951), 236–240.

[3] D. M. Gordon, *A survey of fast exponentiation methods.* Journal of Algorithms **27** (1998), 129–146.

[4] R. L. Graham, A. C.-C. Yao, F.-F. Yao, *Addition chains with multiplicative cost.* Discrete Math. **23** (1978), 115–119.

[5] A. F. Horadam, *Jacobsthal representation numbers.* Fibonacci Quart. **34** (1996), 40–54.

[6] D. E. Knuth, *The Art of Computer Programming* **2**: *Seminumerical Algorithms* (third edition), Reading: Addison Wesley, 1998.

[7] Neal Koblitz, *CM-curves with good cryptographic properties*, in: Feigenbaum (ed), *Advances in Cryptology — Proceedings of Crypto '91*, Lecture Notes in Computer Science **576**, (1991), 279–296.

[8] D. P. McCarthy, *Effect of improved multiplication efficiency on exponentiation algorithms derived from addition chains.* Math. Comp. **46** (1976), 603–608.

[9] F. Morain, J. Olivos, *Speeding up the computations on an elliptic curve using addition-subtraction chains.* RAIRO Inform. Theory **24** (1990), 531–543.

[10] N. J. A. Sloane, S. Plouffe, *The encyclopedia of integer sequences.* San Diego: Academic Press, 1995. `http://www.research.att.com/ njas/sequences/`

[11] Hugo Volger, *Some results on addition/subtraction chains.* Information Processing Letters **20** (1985), 155–160.

Wieb Bosma
Vakgroep Wiskunde
Universiteit van Nijmegen
The Netherlands
E-mail : `wieb@sci.kun.nl`