# The Crouton Algorithm for Optimal Addition Chains

Neill Clift*

April 5, 2024

**Abstract**

We describe a new algorithm for calculating optimal addition chains that has very good performance for single values. The algorithm performs well with or without a database of previously created optimal addition chain lengths. Improved performance comes from the realization that the end of an addition chain can be viewed as a vector addition chain. The values generated by the start of the addition chain together with the values from the trailing vector addition chain satisfy a linear Diophantine equation. By broadening the search tree by differentiating cases based on which constructed chain elements will be used in the future, a collection of new pruning techniques are possible. Estimates of the minimum Hamming weight of solutions to a linear Diophantine equation along with bounds propagation techniques are used to prove that partially constructed chains will not be successful and hence can be pruned.

# Introduction

An *addition chain* is a finite sequence of integers (we call *elements*) $1 = a_0 < a_1 < \ldots < a_r = n$ with $a_i = a_j + a_k$, $i > j \geqslant k \geq 0$ for a *target* $n$ of *length* $r$. We will refer to an addition chain as *formal* if we explicitly specify how each element is constructed. For example the chain $1, 2, 3, 4$ has two ways of constructing 4 since $4 = 2 + 2 = 1 + 3$. We denote with $l(n)$ the length of the *optimal* (shortest) addition chain for $n$. We will call each $a_i$, $i > 0$ a *step*. We define $\lambda(n) = \lfloor log_2(n) \rfloor$ and use it to partition the steps into two cases. If $\lambda(a_i) = \lambda(a_{i-1})$ we say it is a *small* step. Otherwise we must have $\lambda(a_i) = \lambda(a_{i-1}) + 1$ and call it a *large* step. A step of the form $a_i = a_{i-1} + a_j, i > j$ is called a *star* step. If we denote with $s(n)$ the number of small steps in an optimal chain for $n$ we must have $s(n) = l(n) - \lambda(n)$. We will denote with $v(n)$ the binary digit sum of $n$. The smallest target with an optimal addition chain of length $r$ is denoted with $c(r)$. Knuth [11] notes that proving via computer

*NeillClift@live.com

searches that $l(c(r)) > r-1$ is particularly time consuming. On the other hand finding an addition chain of length $r$ for $c(r)$ appears quite fast in comparison. Scholz and Brauer [13, 3] conjectured (Scholz-Brauer conjecture) and partially proved that $l(2^n-1) \leq l(n)+n-1$. Stolarsky [14] noted that he saw only equality in all values he could prove ($l(2^n-1) = l(n)+n-1$). Knuth and Stolarsky [10, 14] conjectured (Knuth-Stolarsky conjecture) that $v(n) \leq 2^{s(n)}$. This conjecture has been proved for $s(n) \leq 3$ [16, 7] and that $v(n) > 8$ requires $s(n) \geq 4$.

# History of Computer Calculations

Knuth appears to have written the first computer program for calculating optimal addition chains in December 1963 (private communication). He used a program written in Algol that ran on the Burroughs B5000 computer and calculated all optimal addition chains for $n \leq 1024$. He did further calculations on the CDC 6600 in assembler [12]. He worked backward from the target splitting $n = p+q$, $p \geq q$ and used memoization of prior calculated $l(n)$ values. The properties of the function $l(n)$ are very interesting and often counter intuitive. As each new technique is developed to calculate optimal chains for larger targets new properties often emerge. The first computer calculations for example found $l(n) = l(2n)$ with $n = 191$. Kato [9] found minimum length merges of addition chains for $n-k$ and $k$ with $1 \leq k \leq \frac{n}{2}$ as well as enumerations of partitions for $n-1$ to find addition chains for $n$. A breadth first search (BFS) was used by Chin and Tsai [5] to find optimal addition chains. They attempted to prune by having all early steps being doubles but discovered that there are only two optimal chains for 2617 and both include the element 5. Tsai [17] found optimal chains for batches of numbers and forced the construction of the final element in a chain to be a star step. Brlek and Mallette [4] calculated batches of optimal addition chains and used bounds for addition chain elements. Using the nomenclature developed by Thurber [15] these would be the most simple class 1 (sequence A) bounds so that an element $a_i$ for an optimal addition chain for $n$ of length $r$ is bounded by $a_i \geq \left\lceil \frac{n}{2^{r-i}} \right\rceil$. Bleichenbacher and Flammenkamp [8] made huge strides by calculating all $l(n)$ for $n \leq 2^{22}$ by using graph reductions and memoization of computed $l(m), m < n$ values for computing $l(n)$. Thurber [15] developed a set of bounds for the elements of an addition chain and the sum of two consecutive elements. These can be used in an addition chain algorithm when elements are selected to cut down the search space. They can also be used to work out the minimum number of steps between two given values in an addition chain. Bahig [1] outlined conditions where steps must be star and limits on the types of non-star steps and hence cut down the time needed to generate branches in the search tree. Clift [6] calculated batches of addition chains using graph enumeration avoiding sub-graphs proven to not be present in optimal chains.

# Broadening the Search Tree

Prior work has shown that the end of an addition chain can be viewed as a vector addition chain that consumes some of the earlier values to get to the target. We can show that the number of elements from the start of the chain that are consumed must be small and related to the small step count of the target. With this in mind we will describe an algorithm that splits the problem such that a small packet of numbers (called a crouton) is selected at each stage that must be used by the chain in the future. A crouton can also be thought of as the temporary variables needed to compute the rest of the chain. This will make the search tree broader but enable new pruning techniques. We will then show how to bound the values in the vector addition chain such that many cases can be rejected early.

# Recursive Algorithm Outline

We now describe the initial state and recursive transformation the new algorithm uses to traverse the search space. We will be searching for an optimal addition chain $A = b_0 < b_1 < \ldots < b_d = n$ for $n$ of length $d \leq l(n)$. Obviously when $d < l(n)$ the search for a chain will fail. We will say we are at depth $r$ with $0 \leq r \leq d$ and mean we have generated the sequence $1 = b_0 < b_1 < \ldots < b_r$. At depth $r$ though we will not keep or use all $b_i, 0 \leq i \leq r$ but instead we will have chosen to use a subset of these values $a_1 < a_2 < a_z$ with $z \geq 1$. There must be a mapping $\alpha$ such that $a_i = b_{\alpha(i)}$. Each $a_i < n$ must be consumed in at least one addition chain step $\beta(i) > r$ so $b_{\beta(i)} = a_i + b_j$ with $j < \beta(i)$.

### Initial State

We start with a single crouton $< 1 >$. So $a_1 = b_0 = 1$ and $z = 1$.

### Recursive Transformation

If we start with a crouton $< a_1, \ldots, a_z >$ we form a new element $a_{z+1} = a_i + a_j$, $i \geq j$. From this we potentially can form up to four new croutons to search:

$< a_1, \ldots, a_{z+1} >$The new crouton has $z + 1$ elements.

$< a_1, \ldots, a_{i-1}, a_{i+2}, \ldots, a_{z+1} >$ The new crouton has $z$ elements since element $a_i$ is dropped.

$< a_1, \ldots, a_{j-1}, a_{j+2}, \ldots, a_{z+1} >$ The new crouton has $z$ elements since element $a_j$ is dropped.

$< a_1, \ldots, a_{j-1}, a_{j+2}, \ldots, a_{i-1}, a_{i+2}, \ldots, a_{z+1} >$, $i \neq j$ The new crouton has $z - 1$ elements since elements $a_i, a_j$ are dropped.

### Termination State

The final crouton at depth $d$ must be $< a_1 = n >$ if the search is successful. The initial and termination state along with the fact that the recursive step can only increase or decrease $z$ by at most 1 leads to the following simple bounds:

$$z \leq \min(r+1, d-r+1)$$

## Bounding the Crouton's Size

Let $< a_1, a_2, ..., a_z >$be a crouton with $z \geq 1$. Let these $a_i$ be part of an optimal addition chain within the first $r+1$ values for $n$ with $l(n) = d$. Hence $\{a_i | 1 \leq i \leq z\} \subset A$ where $A$ is an addition chain for $n$. Further we will specify that the members of the crouton are the only elements in the addition chain within the first $r+1$ values that are used in the rest of the chain. We specify that all elements in the crouton must be used later in the chain. From the requirement that $A$ is an optimal chain we must have $l(\{a_1, a_2, \ldots, a_z\}) = r$. The corresponding vector chain that consumes the crouton must have $l([x_1, x_2, \ldots x_z]) = d - r$ and hence $l(\{x_1, x_2, \ldots x_z\}) = d - r - z + 1$. We know that

$$\begin{aligned} n &\leq 2^{d-r-z+1} \sum_{i=1}^{z} a_i \\ &\leq 2^{d-r-z+1} \sum_{i=1}^{z} 2^{r+1-i} \\ &\leq 2^{d-2z+2}(2^z - 1) \end{aligned}$$

From $s(n) = l(n) - \lambda(n)$ we obtain

$$\begin{aligned} s(n) &= d - \lambda(n) \\ &\geq d - \lambda(2^{d-2z+2}(2^z - 1)) \\ &\geq 2z - 2 - \lambda(2^z - 1) \\ &\geq z - 1 \end{aligned}$$

From this we can see that the crouton remains small since $s(n)$ is slow growing.

## The Frobenius Equation

Our two addition sequence problems are linked because the vector addition chain consumes the elements of the crouton to reach the target. So from $l(\{a_1, a_2, \ldots, a_z\}) = r$ and $l(\{x_1, x_2, \ldots x_z\}) = d - r - z + 1$ we know that:

$$\sum_{i=1}^{z} a_i x_i = n \tag{1}$$

We can prune the search tree by proving there are no solutions to the Frobenius equation when limited by the complementary addition sequence problem. The most obvious way of eliminating cases uses the GCD:

$$GCD(a_1, a_2, \ldots, a_z)|n$$

The many divisions needed to calculate the GCD make this test better done after the following bounds on $x_i$. For example we must have

$$1 \leq x_i \leq 2^{d-r-z+1}$$

We can do much better than this though if we can estimate a lower bound for $s(\{x_1, x_2, \ldots, x_z\})$. We can do this by getting a lower bound for

$$\max_{i=1}^{z} v(x_i)$$

We start with

$$\begin{aligned} v(n) =& v\left(\sum_{i=1}^{z} a_i x_i\right) \\ \leq& \sum_{i=1}^{z} v(a_i x_i) \\ \leq& \sum_{i=1}^{z} v(a_i) v(x_i) \\ \leq& \max_{i=1}^{z} v(x_i) \sum_{i=1}^{z} v(a_i) \end{aligned}$$

So we have

$$\max_{i=1}^{z} v(x_i) \geq \left\lceil \frac{v(n)}{\sum_{i=1}^{z} v(a_i)} \right\rceil \tag{2}$$

From both theory and computational results available via pre-computed $l(n)$ we have:

$$s(x) \geq s_l(x) = \begin{cases} \lceil log_2(v(x)) \rceil & v(x) \leq 64 \\ 7 & v(x) > 64 \end{cases}$$

Immediately we can see that

$$1 \leq x_i < 2^{d-r-z+2-\max_{i=1}^{z} s_l(x_i)}$$

For $\max_{i=1}^{z} s_l(x_i) \leq 6$ we have $\max_{i=1}^{z} v(x_i) \leq 2^{\max_{i=1}^{z} s_l(x_i)}$. We can therefor refine the limit. The maximum value in a chain of length $l$ with at least $s = \max_{i=1}^{z} s_l(x_i)$ small steps must be:

$$\begin{aligned} x_i \leq & 2^{l-s} + 2^{l-s-1} + \cdots + 2^{l-s-2^s+1} \\ \leq & 2^{l+1-s-2^s}(2^{2^s} - 1) \end{aligned}$$

$$1 \leq x_i \leq 2^{d-r-z+2-\max_{i=1}^{z} s_l(x_i) - 2^{\max_{i=1}^{z} s_l(x_i)}}(2^{2^{\max_{i=1}^{z} s_l(x_i)}} - 1) \tag{3}$$

Many cases are rejected in practice by applying this bound to the original Frobenius equation:

$$n \leq \left(\max_{i=1}^{z} x_i\right)\left(\sum_{i=1}^{z} a_i\right) \tag{4}$$

Practically this inequality is not satisfied in a significant proportion of cases and search times are reduced considerably. This pruning will be made clearer with an example.
When trying to find an optimal chain for 42833 of length 19 (which must succeed since computer calculations show $l(42833) = 19$) we get to a depth $r = 8$ with the crouton$< 5, 32, 67 >$. We could have gotten to this point in the search with the partial addition chain $1, 2, 3, 5, 8, 16, 32, 64, 67$.
From 1 we have the following Frobenius equation to solve:

$$5x_1 + 32x_2 + 67x_3 = 42833$$

From 2 $\max_{i=1}^{z} v(x_i) \geq \left\lceil \frac{v(n)}{\sum_{i=1}^{z} v(a_i)} \right\rceil = 2$. So at least one $x_i$ must have $v(x_i) \geq 2$ so that $\max_{i=1}^{z} s_l(x_i) \geq 1$ and hence that $s(\{x_1, x_2, x_3\}) \geq 1$ and so from 3 $x_i \leq 384$. We can immediately abandon this search since from 4 we have the contradiction $42833 \leq 384(5 + 32 + 67) = 39936$.

## Better Estimates for $\max_{i=1}^{z} v(x_i)$

Bounding the size of the $\max_{i=1}^{z} v(x_i)$ has such a large effect on performance that trying to get a better estimate of $\max_{i=1}^{z} v(x_i)$ can be worthwhile even it it takes a non-trivial amount of computation.
If we say that $n \mod m$ represents the least non-negative element from the residue class modulo $m$. It is curios that:

$$v(n \bmod (2^i - 2^j)) \leq v(n),\ i > j \tag{5}$$

To prove this we will assume a smallest counter example $v(r) > v(n)$ with $r = n \bmod (2^i - 2^j)$. We must have $n > r$. We will first assume that the binary representation of $n$ contains a power of two ($2^{i+k}$, $k \geq 0$). Since $2^{i+k} \equiv 2^{i+k} -$

$(2^{i+k} - 2^{j+k}) \pmod{(2^i - 2^j)}$. We have a contradiction since $v(n - 2^{i+k} + 2^{j+k})$ would be a smaller counter example from the residue class:

$$\begin{aligned} v(n - 2^{i+k} + 2^{j+k}) \leq & v(n - 2^{i+k}) + v(2^{j+k}) \\ \leq & v(n) - 1 + 1 \\ \leq & v(n) \\ < & v(r) \end{aligned}$$

We may therefore assume that $2^i - 2^j < n \leq 2^{i-1} - 1$. The binary representation of $n$ must be $2^{i-1} + 2^{i-2} + \cdots + 2^j + \cdots$ and we obtain the following contradiction:$v(n - (2^i - 2^j)) < v(n) < v(r)$ since $n - (2^i - 2^j) < n$.
From this we obtain the following inequality:

$$\max_{i=1}^{z} v(x_i) \geq \max_{i=1}^{z} v(x_i \bmod m) \geq \left\lceil \frac{v(n \bmod m)}{\sum_{i=1}^{z} v(a_i \bmod m)} \right\rceil, \; m = 2^j - 2^k, \; j > k \tag{6}$$

Proof:

$$\begin{aligned} n \bmod m = & (\sum_{i=1}^{z} a_i x_i) \bmod m \\ = & (\sum_{i=1}^{z} (a_i \bmod m)(x_i \bmod m)) \bmod m \\ v(n \bmod m) = & v((\sum_{i=1}^{z} (a_i \bmod m)(x_i \bmod m)) \bmod m) \\ \leq & v(\sum_{i=1}^{z} (a_i \bmod m)(x_i \bmod m)) \\ \leq & \sum_{i=1}^{z} v(a_i \bmod m) v(x_i \bmod m) \\ \leq & \max_{i=1}^{z} v(x_i \bmod m) \sum_{i=1}^{z} v(a_i \bmod m) \end{aligned}$$

This is a generalization of the realization that the bottom $k$ ($2^k = 2^{k+1} - 2^k$) bits of $n$ can be determined only by the bottom $k$ bits of $a_i$ and $x_i$. This will be clear in the following example. With the crouton $< 5, 8 >$ at depth $r = 4$ while trying to find an addition chain for $n = 35$. From 2 we would have $\max_{i=1}^{z} v(x_i) \geq \left\lceil \frac{v(n)}{\sum_{i=1}^{z} v(a_i)} \right\rceil = 1$. If we instead look at the bottom two bits ($k = 2$) then $\max_{i=1}^{z} v(x_i) \geq \left\lceil \frac{v(n \bmod 2^k)}{\sum_{i=1}^{z} v(a_i \bmod 2^k)} \right\rceil = \left\lceil \frac{2}{1} \right\rceil = 2$. This window method will also detect in-feasibility when $2^k | GCD(a_1, a_2, \ldots, a_z)$ but $2^k \nmid n$.

An example from the more general case has a crouton of $< 25, 84 >$at depth $r = 8$ while searching for an addition chain for $n = 42163$ of length $d = 19 < l(42163) = 20$. This search will fail. We could have encountered this situation via the partial chain $1, 2, 4, 8, 16, 17, 25, 42, 84$. With $m = 12$ we get:

$$\max_{i=1}^{z} v(x_i) \geq \left\lceil \frac{v(n \bmod 12)}{\sum_{i=1}^{z} v(a_i \bmod 12)} \right\rceil = \left\lceil \frac{3}{1} \right\rceil = 3$$

So we must have $s(\{x_1, x_2\}) \geq 2$ and so from $3x_i \leq 480$. This is not enough to prune this path completely since $(25 + 84)480 > 42163$ but it limits the possibilities for the $x_i$.
We may obtain a similar bound by multiplying both sides of 1 by a constant $m$:

$$\max_{i=1}^{z} v(x_i) \geq \left\lceil \frac{v(mn)}{\sum_{i=1}^{z} v(ma_i)} \right\rceil \tag{7}$$

Practically only $m = 3$ seems to allow performance improvements. For example with the crouton $< 34, 43, 86 >$ at depth $r = 9$, $n = 58951$ and $d = 20$ we obtain:

$$\max_{i=1}^{z} v(x_i) \geq \left\lceil \frac{v(3n)}{\sum_{i=1}^{z} v(3a_i)} \right\rceil = \left\lceil \frac{10}{8} \right\rceil = 2$$

So we must have $s(\{x_1, x_2\}) \geq 1$ and so from $3x_i \leq 384$.

## Enumerating Solutions

For problems that appear small it helps the performance of the algorithm to attempt to enumerate all the solutions to the Frobenius equation and attempt to eliminate them by finding a lower bound for $l(\{x_1, ..., x_z\})$. Knowing the bounds of $x_i$ and also the bounds of $s(\{x_1, ..., x_z\})$ enumerating problems with values for each $l_i \leq x_i \leq u_i$ with $u_i - l_i \leq\sim 300$ seems to work well. Trying to solve the Frobenius equation directly using the extended GCD algorithm is too expensive. Presumably because of the many divides needed. Improving $l_i, u_i$ via the standard techniques of constraint satisfaction appears to be too slow as well. The best method found is to use a depth first search starting with the $x_i$ corresponding to the largest $a_i$.
We will follow an example to see the sort of pruning done. We have the crouton $< 16, 17, 67 >$ at depth $r = 8$ for an addition chain for $n = 152557$ of length $d = 21 < l(152557) = 22$. So this search will eventually fail. We have $a_1 = 16$, $a_2 = 17$ and $a_3 = 67$. Simple bit counting has $\left\lceil \frac{v(152557)}{v(16)+v(17)+v(67)} \right\rceil = 2$. So $u_i \leq 1536$ and $l_i \geq 1$. Using 4 we know that $s(\{x_1, x_2, x_3\}) = 1$ and hence $v(x_i) \leq 2$. With these bounds we can compute the initial bounds for $x_3$:

$$1521 \leq x_3 \leq 1536$$
$$1536 \leq x_3 \leq 1536 \text{ (using } v(x_3) \leq 2)$$

The value $x_3 = 1536$ can immediately be dismissed since $v(152557-67.1536) = 9$ and $v(a_1x_1 + a_2x_2) \leq 6$. If this were not the case we would set $a_3 = 1536$ and repeat the bounding process for $a_2$ with $n = 152557 - 67.1536$. Pruning techniques like the binary digit count of $x_i$ work well because they can eliminate a large number of possibilities quickly. This examples shows another way to reduce the range via the most significant bit of $u_3 = 1536$. If we assume that $x_3 \geq 2^{\lambda(u_3)}$ then

$$\begin{aligned} 6 &\geq v(n - a_3x_3) \\ &= v(n - 2^{\lambda(u_3)}a_3 - a_3(x_3 - 2^{\lambda(u_3)})) \\ &\geq v(n - 2^{\lambda(u_3)}a_3) - v(a_3)(v(x_3) - 1) \\ &\geq 9 \end{aligned}$$

So $x_3 < 1024$. This process can obviously be repeated for the new most significant bit of $u_3$ if there is still a valid range for $x_3$.

### Results

To see the potential of this pruning technique in searching for addition chains I took the code used by Ed Thurber to investigate the number of optimal addition chains for $n$ defined as $NMC(n)$. The code had a simple stack of numbers representing the end of a partial addition chain at various depths. New elements are pushed onto the stack by summing all previous elements and pushing them on the stack if they satisfy the addition chain bounds. Duplicates are removed. The code was changed to push croutons on a stack and prune not only with the addition chain bounds but also the bounds described by 4, 2, 6 with modulo $2^k$ and 7 with the multiplier of 3. The new code searches for formal addition chains were chain construction is differentiated by how elements are constructed as well as their value. So we must filter out duplicates. Despite this we see a big improvement in performance as show in table .
Bahig [2] conjectured that there is always an optimal addition chain for $n$ with the last $\left\lfloor \frac{l(n)}{2} \right\rfloor$ steps star. The first counter example is 5979345 whose 16 optimal addition chains of length 27 each have two small steps:
1 2 4 8 16 17 32 49 81 162 324 648 1296 2592 5184 5201 10368 20736 25937 46673 93346 186692 373384 746768 1493536 2987072 2992273 5979345

## References


[1] Hatem M Bahig. Improved generation of minimal addition chains. *Computing*, 78(2):161–172, 2006.

Table 1: Times to Calculate NMC values

| $n \leq$ | Old Seconds | New Seconds |
|---|---|---|
| $2^8$ | 0.3 | 0.2 |
| $2^9$ | 2.8 | 1.1 |
| $2^{10}$ | 26.8 | 6.0 |
| $2^{11}$ | 251.1 | 32.8 |
| $2^{12}$ | 2676.0 | 199.4 |
| $2^{13}$ | 27248.6 | 1226.5 |

Table 2: Effects of different Pruning Techniques

| $n$ | STN | CR | CR+G | CR+GB | CR+GBD |
|---|---|---|---|---|---|
| 5979345 All chains | 1.1 | 1.7 | 0.7 | 0.1 | <0.1 |
| $s(375494703) > 6$ | 554.4 | 886.1 | 310.5 | 29.3 | 5.2 |
| $s(c(27) = 2211837) > 5$ | 6.3 | 10.2 | 4.1 | 0.6 | 0.1 |
| $s(c(30) = 14143037) > 6$ | 49.8 | 85.3 | 30.3 | 4.6 | 0.8 |
| $s(c(32) = 46444543) > 6$ | 292.3 | 483.8 | 168.5 | 16.0 | 2.5 |
| $s(6580895885) > 6$ | 549.4 | 945.0 | 286.2 | 33.1 | 6.2 |
| $s(6442718499) > 6$ | 714.3 | 1158.7 | 383.8 | 38.4 | 7.3 |
| $s(6442517597) > 6$ | 652.0 | 1125.3 | 336.1 | 27.4 | 5.3 |
| $s(2^{31} - 1) > 6$ | 53.9 | 100.5 | 29.7 | 0.3 | 0.1 |
| $s(2^{35} - 1) > 6$ | 100.5 | 188.5 | 52.6 | 0.5 | 0.1 |
| $s(2^{37} - 1) > 6$ | 135.2 | 257.2 | 68.6 | 0.2 | 0.1 |
| $s(2^{39} - 1) > 6$ | 179.4 | 340.6 | 112.0 | 0.1 | <0.1 |
| $s(2^{41} - 1) > 6$ | 234.7 | 449.7 | 113.1 | <0.1 | <0.1 |

STN - Single value algorithm [6]
CR - Crouton, +G with GCD pruning, +B bounds 4 +D memoized $l(n)$ data.

Table 3: Time vs. Binary Digit Sum

| $n$ | CR+GE | CR+GED |
|---|---|---|
| $s(2^{47} - 1) > 7$ | 104.4 | 36.8 |
| $s(2^{53} - 1) > 7$ | 61.9 | 20.6 |
| $s(2^{55} - 1) > 7$ | 60.2 | 21.7 |
| $s(2^{57} - 1) > 7$ | 41.2 | 20.6 |
| $s(2^{58} - 1) > 7$ | 48.0 | 16.7 |
| $s(2^{59} - 1) > 7$ | 39.9 | 35.8 |
| $s(2^{61} - 1) > 7$ | 31.8 | 30.5 |
| $s(2^{62} - 1) > 7$ | 37.6 | 14.5 |
| $s(2^{63} - 1) > 7$ | 33.4 | 16.1 |

[2] Hatem M Bahig. Star reduction among minimal length addition chains. *Computing*, 91(4):335–352, 2011.

[3] Alfred Brauer. On addition chains. *Bulletin of the American Mathematical Society*, 45(10):736–739, 1939.

[4] R. Brlek, S.;Mallette. Sur le calcul des chaÃ®nes d'additions optimales. *Atelier de combinatoire franco-quÃ©becois (6-7 Mai 1991, Bordeaux, France)*, pages 71–85, 1992. Publ. du LACIM 10, ISBN 2-89276-101-8.

[5] YH Chin and YH Tsai. Algorithms for finding the shortest addition chain. In *Proceedings of national computer symposium, Kaoshiung, Taiwan, December*, pages 1398–1414, 1985.

[6] Neill Michael Clift. Calculating optimal addition chains. *Computing*, 91(3):265–284, March 2011.

[7] A. Flammenkamp. Drei beitrÃ€ge zur diskreten mathematik: Additionsketten, no-three-in-line-problem, sociable numbers, 1991.

[8] D. Bleichenbacher; A. Flammenkamp. An efficient algorithm for computing shortest addition chains. 1997.

[9] H. Kato. *On Addition Chains*. PhD thesis, University of Southern California, 1970.

[10] Donald E Knuth. *The art of computer programming, 2: seminumerical algorithms, Addison Wesley*. 1969.

[11] Donald E Knuth. *The art of computer programming, 2: seminumerical algorithms, Addison Wesley*. 1998.

[12] Donald Ervin Knuth. Calculations on addition chains. 1969.

[13] Arnold Scholz. Aufgabe 253 (problem 253). *Jahresbericht der deutschen Mathematiker-Vereinigung*, 47(2):41–42, 1937.

[14] Kenneth B Stolarsky. A lower bound for the scholz-brauer problem. *Canad. J. Math*, 21:675–683, 1969.

[15] Edward G. Thurber. Efficient generation of minimal length addition chains. *SIAM Journal on Computing*, 28(4):1247–1263, 1999.

[16] E.G. Thurber. *The Scholz-Brauer Problem on Addition Chains*. PhD thesis.

[17] Y.H. Tsai. A study on some addition chain problems. Master's thesis, National Tsing-Hua University, Hsinchu, Taiwan, 1985.