Exploring Computational Complexity and Properties of a Natural Class of Addition Chains

Preprint · July 2023

DOI: 10.13140/RG.2.2.22119.83361

1 author:

El Houssaine Chahboun
Mohammed VI Polytechnic University
2 PUBLICATIONS **0** CITATIONS

Addition Chains

Exploring Computational Complexity and
Properties of a Natural Class of Addition Chains

El Houssaine CHAHBOUN

July 2023

Abstract

Calculating the power of a number or any mathematical object according to a given exponent is a common task in various scenarios. Addition chains are a mathematical concept that can elegantly explain the process of exponentiation. This document delves into a specific natural class within the domain of Addition Chains, referred to in this text as the Elho Addition Chains class. By examining the computational complexity of this class and providing theoretical proofs, this document uncovers intriguing properties associated with these addition chains and their role in efficient exponentiation. Additionally, this document presents a result pertaining to number theory, which may be of interest to those who are especially interested in arithmetics and Euclidean division.

Contents

1 Introduction

1.1 Definition

An addition chain for a positive integer n is a set $1 = a_0 < a_1 < \ldots < a_r = n$ of integers such that for each $i \geq 1$, $a_i = a_j + a_k$ for some $k \leq j < i$. The length of an addition chain represents the number of additions required to express all the numbers, which is one less than the total count of numbers in the sequence (cardinality), This length[2] also clearly represents the number of multiplication steps $X^{a_m} = X^{a_i} * X^{a_j}$ in a corresponding calculation of X^n. For the sake of simplicity, we will refer to the length of an addition chain as its cardinality.

2 Elho Addition Chains

2.1 Construction :

Addition chains can also be constructed in the following way, given the integer n:
Choose r < n, which is coprime with n, and start with the set $S := \{n, r\}$. At each step, add to S the difference between the two smallest elements of S, and stop when 0 has been added to S.

Algorithm 1 Elho Addition Chain

Require: $n > r \geq 1$
Ensure: S
  $S \leftarrow \{n\}$
  $N \leftarrow n$
  while $r \neq 0$ **do** ▷ Does this loop end? how many iterations?
    $N \leftarrow N - r$
    $N, r \leftarrow max(N, r), min(N, r)$
    $S \leftarrow S.append(N)$
  end while

It is worth noting that $(r_k)_k$ defined in the algorithm exhibits both positivity and strict monotonic decrease and therefore reaches zero.

2.2 Complexity :

In this section, we will present a formula that determines the algorithm's number of steps, along with its corresponding best-case and worst-case bounds.

2.2.1 Intermediary results:

(1) The sequence $(r_k)_k$ defined in the Elho Addition Chains algorithm traverses the sequence of remainders $(r'_k)_k$ in the Euclidean algorithm, descending in value (although not strictly).

(2) The set S returned by the Elho Addition Chains algorithm is an addition chain.

(3) $card(\{k, r_k = r'_i\}) = \lfloor r'_i / r'_{i-1} \rfloor$

(4) The Euclidean algorithm always needs less than $O(log_\phi(b))$ steps to find the GCD of $a > b > 0$.

Proof: in this proof we will be using the usual version of euclidean division.

(1) : Without loss of generality let's suppose that S_k which reprents the set S at the k'th iteration is ordered decreasingly, notice that r_k is the smallest element of S_k : we have $S_0 = \{n\}; r_0 = n$ and $S_1 = \{n, r\}; r_1 = r$. Let's prove by induction that $(r_k)_k$ travers decreasingly all the terms of the sequence of rests in the euclidean algorithm (plus zero).

Assuming the result up to rank $k \geq 1$,by selecting a suitable k (minimal) we can assume that $r_k = r'_i$ with $i \geq 1$ and that $r_{k-1} = r'_{i-1}$, this implies that the second smallest element of S_k is none other than r'_{i-1} and therefore the elements to be appended in the next q_i iterations, with q_i is the quotient of the usual euclidean division of (r'_{i-1}) by (r'_i) are respectively :

$(r'_{i-1} - 1 * r'_i > ... > r'_{i-1} - q_i * r'_i = r'_{i+1})$. And since $r'_i - r'_{i-1} = (q_i - 1)r'_{i-1} + r'_{i+1}$ by definition of $(r'_i)_i$ only the q_i'th element appended is lesser than (r'_i) and therefore for the $(q_i - 1)$ **(*)** next iterations the smallest element is equal to r_k which is, by the induction hypothesis, the ith rest in the sequence of rests of the euclidean algorithm and the (k+q)'th element is equal to (r'_{i+1}) and therefore we prove the result.

(2) **(*)**and **(1)** implie that the elements of S are in the form of : $r_{i-1} - h * r_i$ with $h \leq q_i - 1$ which can be written as a sum of two previous elements**(if S is ordered increasingly)** of s as follows :

if $h < q_i - 1$: $r_{i-1} - h * r_i = r_{i-1} - (h+1) * r_i + r_i$ | if $h = q_i - 1$: $r_{i-1} - h * r_i = r_{i+1} = r_i - r_{i+2} + r_{i+2}$

(3) is a direct consequence of **(*)**.

(4) ...

Theorem 1 *The number of iterations of the Elho Addition Chains algorithm verifies:*

$$N_{iterations} = \sum_{i=1}^{N} \left\lfloor \frac{r'_{i-1}}{r'_i} \right\rfloor$$

$\left\lfloor \frac{r'_{i-1}}{r'_i} \right\rfloor$*: the quotient of* r'_{i-1} *divided by* r'_i.
$N_{iterations}$*: the number of iterations of the Elho Addition Chains algorithm verifies.*
r'_i*: the sequence of remainders in the Euclidean algorithm of* n *and* r.
N*: the number of steps in the Euclidean algorithm.*

Proof: $\mathrm{N_{iterations}} = card(S) = card(\mathrm{k},\ \mathrm{r}_k \neq 0) = \sum_{i=1}^{N} card(\{k, r_k = r'_i\}) = \sum_{i=1}^{N} \left\lfloor \frac{r'_{i-1}}{r'_i} \right\rfloor$

Central Euclidean Division & Algorithm:

For every $a \geq b \in Z$ there exist $q \in Z$ and r such that: $a = bq + r$ and $|r| \leq a/2$

We can define a different version of the euclidean algorithm using this division which we will be referring to as "Central Eucliden algorithm" and we will be denoting by "$(r^*_k)_k$" the sequence starting with {n,r} and verifies r^*_{i+2} is the rest of the **central division** of $|r^*_i|$ by $|r^*_{i+1}|$

***!!! We can easily show that the intermediatry results (1) and (3) still hold if we replace the sequence $(r'_k)_k$ with the seuqence $(|r^*_k|)_k$**

Theorem 2 *With the same notations as previously, The number of iterations of the Elho Addition Chains algorithm verifies:*

$$N_{iterations} = \sum_{i=1}^{N} \left\lfloor \frac{|r^*_{i-1}|}{|r^*_i|} \right\rfloor$$

Where $(r^*_k)_k$ *starts with {n,r and verifies* $r^*_{i+2} \equiv /r^*_i| \mod /r^*_{i+1}|$.

2.3 Properties :

2.3.1 Worst case :

on a:

$$\lfloor r_{i-1}/r_i \rfloor \leq r_{i-1}/r_i \leq r_{i-1} \leq n$$

donc

$$\sum_{n=1}^{N} \lfloor r_{i-1}/r_i \rfloor \leq \sum_{n=1}^{N} r_{i-1}/r_i \leq \sum_{n=1}^{N} r_{i-1} \leq \sum_{n=1}^{N} n = N * n$$

on conclu que

$$N_{iterations} \leq N * n \leq (log_\phi(r) + 1) * n$$

The term on the right-hand side, $\log_\phi(r) + 1$, comes from value of N as discussed in [3], which involves the Fibonacci sequence.

Note: this bound is reached for r=1.

2.3.2 Best case :

To reach the desired lower bound we will consider the slightly different version of the formula from the corollary.

$N_{iterations} \geq 2 * N = 2(log_\phi(r) + 1)$

because $|r^*_i| \leq \frac{|r^*_{i-1}|}{2}$

Note: in this case this elho addition chain is just as good in terms of number of steps as the exponentiation by squaring addition chain.

2.3.3 Number theory result :

Theorem 3 *This study has allowed us to deduce the following result pertaining to number theory linking together the sequence of remainders in the Euclidean algorithm and the slightly modified version based on the "centred" Euclidean division mentioned earlier in this paper, which may be of interest to those who are especially interested in arithmetics and Euclidean division.*

$$\sum_{i=1}^{N} \left\lfloor \frac{r'_{i-1}}{r'_i} \right\rfloor = \sum_{i=1}^{N_2} \left\lfloor \frac{|r^*_{i-1}|}{|r^*_i|} \right\rfloor$$

***Notations :**

N: the number of elements in r'_i.
N_2: the number of elements in $(r^*_k)_k$.
r'_i: the sequence of remainders in the Euclidean algorithm of n and r.
N: the number of steps in the Euclidean algorithm.
r^*_k: $(r^*_k)_k$ starts with {n,r} and verifies $r^*_{i+2} \equiv |\mathrm{r}^*_i| \mod |\mathrm{r}^*_{i+1}|$.

2.4 Exponentiation by Squaring

Algorithm 2 Exponentiation by Squaring

Require: $n \geq 0$
Ensure: $y = x^n$
  $y \leftarrow 1$
  $X \leftarrow x$
  $N \leftarrow n$
  while $N \neq 0$ **do**
    if N is even **then**
      $X \leftarrow X \times X$
      $N \leftarrow \frac{N}{2}$ ▷ The number of iterations is in $O(log_2(n))$
    else if N is odd **then**
      $y \leftarrow y \times X$
      $N \leftarrow N - 1$
    end if
  end while

The Exponentiation by Squaring algorithm can be precisely defined for a given number n by utilizing an associated addition chain $(a^n_i)_i$, where:

To simplify, assume $n \geq 2$.

For even n:

$$a^n_0 = 1 \quad \text{and for } i \in [1, \mu(n) - 1], \quad a^n_i = 2^{k_i}$$

Here, $\mu(n)$ is the count of '1' bits in n's binary representation, and k_i denotes the position of the ith non-zero bit. Moreover, for $1 \leq j \leq \mu(n) - 1$, $a^n_{\mu(n)+j} = a^n_{\mu(n)+j-1} + a^n_{\mu(n)-j}$.

For odd n: $a^n_i = a^{n-1}_i$ and $a^n_{2\mu(n-1)} = 1 + a^n_{2\mu(n-1)-1}$.

Hence, the last term of the sequence equals n. Specifically, if n is even, $a^n_{2\mu(n)-1}$ is n. Similarly, if n is odd, $a^n_{2\mu(n-1)} = 1 + a^{n-1}_{2\mu(n-1)-1} = 1 + n - 1 = n$.

Note that the chain's length is $2\mu(n)$ for even n and $2\mu(n-1) + 1$ for odd n, approximately $2\mu(n)$ in both cases.

In conclusion, the algorithm's complexity is around twice the count of '1' bits in n's binary representation, denoted as $\mu(n)$. This establishes that the algorithm operates with a time complexity of $O(\log_2(n))$.

References

[1] Wikipedia contributors. (2023, May 31). Exponentiation by squaring.

[2] The Art of Computer Programming, by D.E. Knuth (Volume 2, Seminumerical Algorithms, section 4.6.3).

[3] Mollin, R. A. (2008). Fundamental Number Theory with Applications (2nd ed.). Boca Raton: Chapman Hall/CRC. ISBN 978-1-4200-6659-3.