TELKOMNIKA, Vol.11, No.11, November 2013, pp. 6447~6453
e-ISSN: 2087-278X

An Efficient Algorithm for Optimal Addition Chains

Daxin Zhu[1], Xiaodong Wang*[1,2]
[1]Quanzhou Normal University, Quanzhou, China
[2]Fuzhou University, Fuzhou, China
*Corresponding author, e-mail: wangxiaodong@qztc.edu.cn

Abstract

The computational aspects of finding the shortest addition chains for an integer are investigated in this work. Theoretically developed lower and upper bounds for the minimal length of the addition chains for an integer are exploited to construct a subtle pruning function for backtracking algorithm. These techniques are finally combined to build an efficient algorithm for finding the optimal addition chains.

Keywords: *addition chains, lower bounds, upper bounds*

1. Introduction

The encryption and decryption in the RSA scheme consist of the power exponentiation x^n [6]. This operation is very important for prime testing and integer factoring algorithms. Exponentiation has become important as many cryptographic algorithms have this operation at their core. Optimizing exponentiation can have significant impact on the running time of cryptographic algorithms. Let us have a look at the operation. For example, we can use 6 multiplications to calculate x^{23} as follows:

$$x, x^2, x^3, x^5, x^{10}, x^{20}, x^{23}$$

It is not difficult to prove that the calculation of x^{23} for given x needs at least 6 multiplications. Therefore the above calculation sequence is optimal. In this sequence, the power chain $1,2,3,5,10,20,23$ forms an addition chain of the integer 23.

In general, an addition chain for an integer n is defined as the sequence

$$1 = a_0 < a_1 < \cdots < a_r = n$$
$$a_i = a_j + a_k, 0 \le k \le j < i, i = 1,2,\cdots,r$$

The number of calculation steps r is called the length of the addition chain for n. The minimal length for which there exists an addition chain for n is denoted by $l(n)$. An optimal addition chain is one of the addition chain with a shortest length $l(n)$. The optimal addition chains are not necessarily unique and the elements of an addition chain may be formed in more than one way.

Addition chains give a very easy way of computing x^n for given x and n. The optimal addition chain for an integer n gives the least number of multiplications needed to compute x^n. The length of an optimal addition chain for an integer n is usually denoted by $l(n)$. Tables of optimal addition chain lengths have been used to benchmark new algorithms. Many counterintuitive properties of $l(n)$ are known and this makes the subject interesting to study. Positive integer addition on the shortest chain issues, has made , has been a lot of meaningful results [2, 3, 5, 7]. However, most of these studies the theoretical study of mathematics.

Many researches and explorations concentrate on finding a short, not necessary minimal addition chain, while few papers study generating all optimal addition chains [1, 4, 8, 10]. The depth first search algorithms to find optimal addition chains has been improved by

Received April 11, 2013; Revised June 28, 2013; Accepted July 13, 2013

proposing various pruning techniques that cut down the search time for generating optimal chains. a new method based on a directed acyclic graph to find optimal addition chains was proposed recently. We are interested in Thurber's algorithm to generate optimal addition chains. We exploit the upper and lower bounds of the theoretical results to design an efficient algorithm to generate optimal addition chains for given integers.

The organization of the paper is as follows.

In the following 4 sections we describe our presented algorithms and our computational experience with these algorithms. In section 2 we describe the standard backtrack algorithm for finding optimal addition chains and in section 3 an improved iterative deepening search algorithm is presented. In section 4 the algorithm is improved further by a lower bound of the problem and a pruning function is discussed. In section 4 the upper bound and lower bound are combined to speed up the search engine. Some concluding remarks are in section 5.

2. Backtracking Algorithm

The most intuitive algorithm to generate optimal addition chains for a given integer is backtracking algorithm. In the state space tree of the problem, the son node a_{i+1} of the node a_i is composed by $a_j + a_k, k \le j \le i$.

The state space tree depth-first backtracking search algorithm for solving addition chain problems can be described as follows.

Algorithm 1 $Backtrack(step)$

```
1: if $a[step] = n$ then
2:   if $step < best$ then
3:     $best \leftarrow step$
4:     $chain \leftarrow a$
5:   end if
6:   return
7: else
8:   for $i \leftarrow step$ downto $1$ do
9:     if $2*a[i] > a[step]$ then
10:      for $j \leftarrow i$ downto $1$ do
11:        $k \leftarrow a[i]+a[j]$
12:        $a[step+1] \leftarrow k$
13:        if $k > a[step]$ and $k <= n$ then
14:          $Backtrack(step+1)$
15:        end if
16:      end for
17:    end if
18:  end for
19: end if
```

In the state space tree for addition chain problem, each node of level k has at least $k+1$ son nodes, so the number of paths from the root to any node of level k is at least $k!$. Therefore, the size of state space tree is growing exponentially. A standard backtracking algorithm can only generate the optimal addition chains for small integer n.

3. Proposed Algorithm

In the standard backtracking algorithm for generating the optimal addition chains described above, a depth-first search method is used to search the state space tree. It is natural for this technique that the first addition chain found by the algorithm is not necessarily the optimal addition chain.

If we use breadth-first search method to search state space tree, then the first addition chain found by the algorithm must be the optimal addition chain. But the space overhead of this approach too big. The iterative deepening search algorithm can guarantee the first addition chain found by the algorithm is an optimal addition chain, and it does not need too much space overhead. The basic idea of the iterative deepening search algorithm is to control the search depth d in the backtracking algorithm. Beginning from $d=1$, the search depth d is increased by 1 after each backtrack search until an optimal addition chain is found.

The iterative deepening search algorithm for finding optimal addition chains can be described as follows.

```
Algorithm 2 IterativeDeepening
1: $best \leftarrow n+1$
2: $found \leftarrow$ false
3: $d \leftarrow 2$
4: while not $found$ do
5:     $a[1] \leftarrow 1$
6:     $Backtrack(1)$
7:     $d \leftarrow d+1$
8: end while
```

In the algorithm, the search engine $Backtrack(step)$ is modified to control the search depth up to d as follows.

```
Algorithm 3 Backtrack(step)
1: if not $found$ then
2:     if $a[step] = n$ then
3:         { an addition chain is found }
4:         if $step < best$ then
5:             $best \leftarrow step$
6:             $chain \leftarrow a$
7:             $found \leftarrow true$
8:         return
9:         end if
10:    else
```

```
11: if $step < d$ then
12: { control the search depth }
13: for $i \leftarrow step$ downto $1$ do
14: if $2*a[i] > a[step]$ then
15: for $j \leftarrow i$ downto $1$ do
16: $k \leftarrow a[i]+a[j]$
17: $a[step+1] \leftarrow k$
18: if $k > a[step]$ and $k <= n$ then
19: $Backtrack(step+1)$
20: end if
21: end for
22: end if
23: end for
24: end if
25: end if
26: end if
```

4. Results and Discussion

Let $\lambda(n)=\lfloor \log_2 n \rfloor$ and $\nu(n)$ be the number of 1's in the binary representation of n. Theoretically developed lower bounds for the length $l(n)$ of an optimal addition chain for an integer n provide better starting values from which to start the search. It is well known [9] that $\lambda(n)+\lceil \log_2 \nu(n) \rceil$ is a lower bound of $l(n)$. We can use this fact to speed up the search in iterative deepening search algorithm by setting the initial search depth d to $\lambda(n)+\lceil \log_2 \nu(n) \rceil$.

On the other hand, pruning functions also help to speed up the algorithm by pruning the search tree.

Let a_i and a_j be two elements of an addition chain, and $a_i > 2^m a_j$. As the doubling step is the fastest way to increase the value of an element of the addition chain, that is $a_i \le 2a_{i-1}, 1 \le i \le r$, from a_j to a_i needs at least m steps.

If we expect to find an addition chain of n on the level d of the state space tree T, then in the subtree rooted at the node a_i of level i, a necessary condition to find such a chain at level d is $2^{d-i}a_i \ge n$. It can therefore be seen that it is impossible to find an addition chain of n in the level d of the subtree rooted at the node a_i, if $2^{d-i}a_i < n$. In this case, the subtree rooted at a_i can be pruned off.

When n is an odd number, this pruning condition can be strengthened further. In fact, we can assert that the last element $a_r = a_j + a_k, k \le j$ of the shortest addition chain must be an odd number provided n is odd. It can thus be extrapolated that $k < j$, otherwise a_r would be even. It can be deduced from the minimality of r that $j = r-1$. Therefore, $a_r = a_{r-1} + a_k, k < r-1$. If we can find an optimal addition chain at level d, that is $r = d$, then $a_{d-1} + a_{d-2} \ge n$. From $a_{d-1} \le 2a_{d-2}$ we know $3a_{d-2} \ge n$. In this case $6a_{d-3} \ge n$ as $a_{d-2} \le 2a_{d-3}$. In general, for $i = 0,1,\cdots,d-2$ we have $3 \times 2^{d-(i+2)}a_i \ge n$. In other words, when $3 \times 2^{d-(i+2)}a_i < n$, it is impossible to find an optimal addition chain before level d in the subtree rooted at node a_i, and thus the subtree rooted at the node a_i can be pruned off.

This pruning condition can be concluded as follows.

Suppose in the iterative deepening search algorithm for optimal addition chains of the integer n, the current search depth be d. A pruning condition for the subtree rooted at the node a_i of level i is:

$$\begin{cases}\log_2(n/3a_i)+i+2>d & 0\le i\le d-2\\ \log_2(n/a_i)+i>d & d-1\le i\le d\end{cases}$$

It is readily seen that the unique addition chain is $1,2,4,\cdots,2^m$ for integer $n=2^m$. For the case of n is not a power of 2, we can express the integer n as $n=2^t(2k+1),k>0$. For this more general case, we can generalize above pruning condition to a more general condition as follows.

Theorem 1 Suppose in the iterative deepening search algorithm for optimal addition chains of the integer n, the current search depth be d, and the integer n can be expressed as $n=2^t(2k+1),k>0$. A pruning condition for the subtree rooted at the node a_i of level i is:

$$\begin{cases}\log_2(n/3a_i)+i+2>d & 0\le i\le d-t-2\\ \log_2(n/a_i)+i>d & d-t-1\le i\le d\end{cases}$$

Proof.

In the case of $0\le i\le d-t-2$ and $\log_2(n/3a_i)+i+2>d$, we have $3\times 2^{d-(i+2)}a_i<n$. From $i\le d-t-2$ and $t\ge 1$ we know $i\le d-3$. Therefore, at least 3 steps are left to find an addition chain at the level d in the subtree rooted at the node a_i. If $a_m=a_j+a_s, s\le j<m$, $m>i+1$, and a_m is not a doubling node, that is $a_m\ne 2a_{m-1}$, then $k<j$. It follows that $j\le m-1, s\le j-1\le m-2$.

Therefore,

$$\begin{aligned}a_m &\le a_{m-1}+a_{m-2}\\ &\le 2^{(m-1)-i}a_i+2^{(m-2)-i}a_i\\ &=2^{(m-2)-i}(2a_i+a_i)\\ &=2^{(m-2)-i}(3a_i)\end{aligned}$$

It follows from $2^{d-m}a_m\ge n$, the necessary condition to find an addition chain at the level d in the subtree rooted atthe node a_m that

$$\begin{aligned}n &\le 2^{d-m}a_m\le 2^{d-m}2^{m-2-i}(3a_i)\\ &<2^{d-2-i}(3)(n/(3\cdot 2^{d-2-i}))=n\end{aligned}$$

A contradiction.

In the case of a_m being doubling nodes for all $m>i+1$, if an addition chain is found at the level d in the subtree rooted at node a_i, then we have $n=a_d=2^{d-(i+1)}a_{i+1}$. This means $2^{d-(i+1)}$ is divisible by n. On the other hand, it follows from $i\le d-t+2$ that $d-i-1\ge t+1$. Therefore, $2^t(2k+1)\bmod 2^{t+1}=0$. That is $2k+1$ is divisible by 2. This is a contradiction. Therefore, in the case of $0\le i\le d-t-2$ and $\log_2(n/3a_i)+i+2>d$, the subtree rooted at a_i can be pruned off.

The proof for the case of $d-t-1\le i\le d$ is similar.

The proof is complete.

We can further improve the standard backtracking algorithm for constructing the shortest addition chain in several aspects as follows.

(1) Using the iterative deepening search strategy.
(2) Getting an accurate estimation of search depth by the lower bound $lb(n)$ on $l(n)$.
(3) Speeding up the search engine by appropriate pruning function.
(4) Constructing a more accurate upper bound $ub(n)$ on $l(n)$ by the power tree.

When $lb(n) = ub(n)$ the addition chain given by power tree is exactly the optimal addition chain.

When $lb(n) < ub(n)$, the optimal addition chain can be found by following improved iterative deepening search algorithm starting from depth $d = lb(n)$.

The improved iterative deepening search algorithm can be described as follows.

```
Algorithm 4 Search
1: $lb \leftarrow lower(n)$ { $lower(n) = \lambda(n) + \lceil \log_2 \nu(n) \rceil$ }
2: $ub \leftarrow power(n)$ { $power(n)$ constructed by power tree}
3: $t \leftarrow gett(n)$ { $n = 2^t(2k+1), k \geq 1$ }
4: if $lb < ub$ then
5: $found \leftarrow$ false
6: while not $found$ do
7: $a[1] \leftarrow 1$
8: $Backtrack(1)$
9: $lb \leftarrow lb + 1$
10: if $lb = ub$ then
11: $found \leftarrow$ true
12: end if
13: end while
14: end if
```

```
Algorithm 5 Backtrack(step)
1: if not $found$ then
2: if $a[step] = n$ then
3: { an addition chain is found }
4: $best \leftarrow step$
5: $chain \leftarrow a$
6: $found \leftarrow true$
7: return
8: else
9: if $step < d$ then
10: { control the search depth }
11: for $i \leftarrow step$ downto 1 do
```

```
        **if** $2*a[i] > a[step]$ **then**
          **for** $j \leftarrow i$ **downto** $1$ **do**
            $k \leftarrow a[i] + a[j]$
            $a[step+1] \leftarrow k$
            **if** $k > a[step]$ **and** $k <= n$ **then**
              **{a pruning function of Theorem 1}**
              **if not** $prune(step+1)$ **then**
                $Backtrack(step+1)$
              **end if**
            **end if**
          **end for**
        **end if**
      **end for**
    **end if**
  **end if**
**end if**
```

5. Conclusion

In the previous sections, we present an efficient algorithm to solve the addition chain problem. The improved algorithm speeds up the generation of optimal addition chains in three aspects. An iterative deepening search algorithm is exploited to search the optimal addition chains. The upper bound and lower bound of the addition chain problem are combined to speed up the search engine.

The computational experiments demonstrate that the achieved results are not only of theoretical interest, but also that the techniques developed may actually lead to considerably faster algorithm.

Acknowledgements

The authors acknowledge the financial support of Natural Science Foundation of Fujian under Grant No.2013J0101, Science and Technology Key Project of Fujian under Grant No.2012Y0101 and the Haixi Project of Fujian under Grant No.A099.

The authors are grateful to the anonymous referee for a careful checking of the details and for helpful comments that improved this paper.

References

[1] DM Gordon. A survey of fast exponentiation methods. *Journal Algorithms*. 1998; 122: 129-146.
[2] H Bahig. A new strategy for generating shortest addition. *Computing*. 2011; 91: 285-306.
[3] F Bergeron, J Berstel, S Brlek, CDuboc. Addition chains using continued fractions. *Journal Algorithms*. 1989; 10: 403-412.
[4] P Downey, B Leong, R Sethi. Computing sequences with addition chains, *SIAM Journal Comput.* 1981; 10: 638-646.
[5] W Jiang, J Zhang, J Li. A Multiagent Supply Chain Information Coordination Mode Based on Cloud Computing. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2013; 11(11): 61-85.
[6] H Bahig. Improved generation of minimal addition chains. *Computing*. 2006; 78: 161-172.
[7] D Dobkin, RJ Lipton. Addition chain methods for the evaluation of specific polynomials, *SIAM J. Comput.* 1980; 9: 121-125.
[8] MC Neill. Calculating optimal addition chains. *Computing*. 2011; 91: 265-284.
[9] R Monteiro, L Reis, AC Pereira. Humanoid Dynamic Controller. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2012; 10(8).
[10] EG Thurber. Addition chains - an erratic sequence. *Discrete Math.* 1993; 122; 287-305.