UNIVERSITÄT KAISERSLAUTERN

ON THE COMPUTATION OF CARMICHAEL NUMBERS

Andreas Guthmann

Preprint Nr. 218

FACHBEREICH MATHEMATIK

ON THE COMPUTATION OF CARMICHAEL NUMBERS

Andreas Guthmann

Preprint Nr. 218

UNIVERSITÄT KAISERSLAUTERN
Fachbereich Mathematik
Erwin–Schrödinger–Straße
6750 Kaiserslautern

April 1992

On the Computation of Carmichael Numbers

Andreas Guthmann, Fachbereich Mathematik, Universität Kaiserslautern, Pfaffenbergstr. 95, D-6750 Kaiserslautern.

1991 Mathematics Subject Classification: 11Y55

Abstract: We present two effective sieve algorithms suitable for the computation of Carmichael numbers in a given interval. One algorithm was implemented on a vector computer VP100 to find the Carmichael numbers up to 10^{14}. We give some statistics on the numbers found.

Introduction

As is well known the converse of Fermat's theorem is not true in general. If n is a prime number then

$$a^{n-1} \equiv 1 \bmod n \text{ if } (a,n) = 1. \tag{1}$$

But (1) does not imply that n is prime. Positive composite integers n satisfying (1) are called ***pseudoprimes to base*** a (psp(a)). For $a = 2$, the smallest psp(2) is $n = 341 = 11 \cdot 31$ found by Sarrus in 1819. It is readily shown that there are infinitely many psp(a) for each a and it is also possible to give lower bounds for the pseudoprime counting function

$$N_a(x) = |\{n \leq x | n \text{ is psp}(a)\}|,$$

as x tends to infinity. Further information can be found in the fundamental paper by Pomerance, Selfridge and Wagstaff[13].

Even the stronger condition

$$a^{n-1} \equiv 1 \bmod n \text{ for each } a \text{ with } (a,n) = 1, \tag{2}$$

is not sufficient for the primality of n. Composite integers for which (2) holds are called *Carmichael numbers*. The smallest one is $n = 561 = 3 \cdot 11 \cdot 17$. If $C(x)$ denotes the number of Carmichael numbers up to x, it is known [13] that for each $\alpha < 1$

$$C(x) \leq x \exp(-\alpha \log x \log_2 x / \log_3 x), \ x \geq x_0(\alpha).$$

Here $\log_k$ denotes the k-fold iterated logarithm. Contrary to the case of $N_a(x)$ our knowledge about $C(x)$ is much scarcer. For instance, it is not even known whether there are infinitely many Carmichael numbers although this is highly probable.

In view of this fact the computation of Carmichael numbers is of certain interest. In the present investigation we give a sieve algorithm for the computation of Carmichael numbers in a given interval. We describe two versions of it which are stated in section 2. Since the method is well suited for implementation on a vector computer we used it to determine the Carmichael numbers up to 10^{14}. Some remarks on the vector program are given in section 3. The results are discussed in section 4. We there also present some statistics about the finer structure of Carmichael numbers based on our computations. Finally, in section 5 we give a method for the construction of Carmichael numbers.

Tables of Carmichael numbers have been computed among others by Swift [14] and Pomerance, Selfridge and Wagstaff [13]. An attempt to compute the Carmichael numbers up to 10^{12} has also been undertaken by Jäschke [6]. It seems however that his algorithm is not easy to implement.

1. The Structure of Carmichael Numbers

The sieve method to be presented in the next section depends very much on the special structure of Carmichael numbers. For the readers convenience we therefore give in this first paragraph some elementary properties of Carmichael numbers, which are, of course, well known [2].

If n is a positive integer, let $\lambda(n)$ denote the smallest positive exponent m such that $x^m \equiv 1 \bmod n$ for each x with $(x, n) = 1$. It is easy to see that

$$\lambda(p^k) = \varphi(p^k) = p^{k-1}(p-1) \text{ if } p \text{ is an odd prime and } k \geq 1,$$
$$\lambda(2) = 1, \ \lambda(4) = 2, \ \lambda(2^k) = 2^{k-2} \text{ for } k \geq 3.$$

Moreover, if $n = \prod_{i=1}^{t} p_i^{e_i}$ is the prime factor decomposition of n, then

$$\lambda(n) = [\lambda(p_1^{e_1}), \ldots, \lambda(p_t^{e_t})],$$

where [...] denotes the least common multiple. Thus we get

Theorem 1: *n is a Carmichael number if, and only if, $\lambda(n)|n-1$.*

Since $\varphi(p^k)|\lambda(p^k)$ if $p^k|n$ we also see that

Theorem 2: *If n is a Carmichael number, then n is squarefree and $p-1|n-1$ for each prime p dividing n.*

Theorem 3: *Every Carmichael number is odd.*

We also get a very convenient sufficient criterion for an integer to be a Carmichael number:

Theorem 4: *A composite integer n is a Carmichael number if, and only if, it is odd, squarefree, and $p-1|n-1$ for each prime p dividing n.*

Proof: The necessity follows from the theorems above. To show sufficiency, let n be odd, squarefree, and assume $p-1|n-1$ for each $p|n$. Given a with $(a,n)=1$, we have $a^{p-1} \equiv 1(p)$, hence, $a^{n-1} \equiv 1(p)$. By the Chinese remainder theorem, (2) holds, q.e.d.

The next theorem has hitherto despite (because of ?) its simplicity not been stated explicitly. However, it plays a crucial role in our method.

Theorem 5: *Let n be a Carmichael number. Then*

$$2p^2 - p \leq n$$

for each prime p dividing n or, equivalently,

$$p \leq \frac{1}{4}(1+\sqrt{8n+1}).$$

Proof: Since $p|n$ and $p-1|n-1$, the Chinese remainder theorem gives $n \equiv p \bmod p(p-1)$. Hence, n is of the form $n = p + jp(p-1)$. But $j=0$ is impossible and $j=1$ leads to $n=p^2$. Thus $j \geq 2$ and the conclusion follows.

As the example $n = 561, p = 17$ shows, the estimate of Theorem 5 is best possible.

2. The Algorithms

Our algorithms depend on the fact that if a prime p divides a Carmichael number n then $n \equiv p \bmod p(p-1)$. The task is to compute all Carmichael numbers in an interval I of positive integers. Consider for a fixed odd prime p a multiple $pm \in I$. By the above, if $pm \not\equiv p \bmod p(p-1)$, i.e. $m \not\equiv 1(p-1)$, then $n = pm$ is not a Carmichael number and can be deleted ("sifted out").

The first algorithm consists of two stages. In the first ("pass 1") the appropriate pm are sifted out and in the second ("pass 2") we determine those of the remaining $n \in I$, which have all their prime factors $\leq P_0$. Here the bound P_0 is chosen according to Theorem 5.

We formulate the algorithm as follows:

Algorithm A: The ordinary sieve.

Input: $N, M \in \mathbf{N}, I = [N, N+M]$ where $M < 2N(N-1)$.

The integers of I are stored in the array a below.

Output: Carmichael numbers in I.

```
for i := 0 to M do a_i := i + N;
P_0 := (1/4)(1 + sqrt(8(N + M) + 1));
P := set of odd primes p ≤ P_0;
for p ∈ P do
begin (* pass 1 *)
    m := [N/p]; if pm < N then m := m + 1;
    if not odd(m) then m := m + 1;
    r := m mod p − 1;
    while pm ≤ N + M do
    begin
      if r ≠ 1 then a_{pm−N} := 0;
      m := m + 2; r := r + 2 mod p − 1
    end
end (* of pass 1 *)
for p ∈ P do
begin (* pass 2 *)
    Define m as in pass 1;
    while pm ≤ N + M do
    begin
```

$a_{pm-N} := a_{pm-N}/p; m := m + 2$

end

end (* of pass 2 *);

for $i := 0$ to M do if $a_i = 1$ then writeln$(N + i)$;

Note that at the beginning of pass 1 for each $p \in \mathbf{P}$ the integer m is determined such that pm is the smallest odd multiple of p in I. Then pm runs through all odd multiples of p in I and r represents the value of $pm \bmod p - 1$.

To verify the correctness of algorithm A we have to show that $a_i = 1$ if, and only if, $N+i$ is a Carmichael number in I. First, assume $a_i = 1$ after the execution of pass 2. The starting value for a_i is $a_i = N + i$. If $N + i$ were even, $a_i = 0$ or $a_i \geq 2$ after pass 2. Hence, $N + i$ is odd. In pass 2, a_i is divided exactly once by primes p such that $p|N+i$ and $p \leq P_0$. Therefore, $N+i$ is squarefree since $a_i = 1$ by assumption. If $p|N + i$, then $N + i = pm$ where m is odd. Moreover, $a_i = a_{pm-N} \neq 0$, which implies $N + i = pm \equiv 1 \bmod p - 1$. Finally, $N + i$ is composite. Otherwise $N + i = p$. But then $N \leq N + i = p \leq P_0 \leq \frac{1}{4}(1 + \sqrt{8M + 8N + 1})$, or, equivalently, $2N(N - 1) \leq M$, contrary to our hypothesis. Altogether we proved that $N + i$ is odd, squarefree, composite and $p - 1|N + i - 1$for each $p|N + i$. Thus $N + i$ is Carmichael by Theorem 4.

Now let conversely $N + i$ be a Carmichael number in I and $0 \leq i \leq M$. If p is a prime divisor of $N + i$, then $p \leq P_0$ by Theorem 5. Write $N + i = pm$, where m is odd. Then $pm = N + i \equiv 1 \bmod p(p - 1)$ and $a_i = a_{pm-N} \neq 0$ after the execution of pass 1. Hence, $a_i = N + i$ at the beginning of pass 2.

In the execution of pass 2, a_i is divided by each p where $p|N+i$. Since $N+i$ is odd and squarefree we obtain $a_i = 1$ after the termination of pass 2. This finishes the proof of the correctness.

The algorithm as stated above is well suited for computational purposes and can be easily implemented. However, there are several improvements, some of which are discussed in the following remarks.

1) Since every Carmichael number is odd, it suffices to store only the odd integers in I. We then let $a_i = N + 2i + 1$ for $0 \leq i \leq \frac{M-1}{2}$ (M should be odd in this case).

2) The succession of pass 1 and 2 is unimportant. If enough memory space is available, the values of m computed in pass 1 can be used in pass 2. This saves the divisions at the beginning of pass 2.

3) The restrictions on N and M stated at the beginning of algorithm A are not as severe as might be suspected at first sight. In fact, the algorithm is intended to

work for consecutive intervals $I_j = [N_j, N_j + M]$, where $N_{j+1} = N_j + M + 1$. This leads to the "segmented sieve" which we now discuss. In this application M is fixed (say $M = 10^6$) and N grows monotonically so that the condition $M < 2N(N-1)$ is satisfied if N is sufficiently large.

The algorithm gains its full effectivity if it is applied to a large interval $I = [N_0, N_0 + k(M+1) - 1]$ that is split into k subintervals $I_j = [N_j, N_j + M]$, say, where $N_j = N_0 + j(M+1)$ for $0 \le j \le k-1$. Then $I = \bigcup_{j=0}^{k-1} I_j$ is a disjoint union of the I_j. This construction (the "segmented sieve") avoids most of the divisions in pass 1, since the last values of m for I_j may the stored and used in I_{j+1}. This idea was also described by Bays and Hudson [1] in their implementation of the sieve of Eratosthenes. The appropriate generalization of algorithm A is straightforward. We leave the details to the reader since we now discuss an even better method, which uses no division at all (execept for initialisation).

The most time consuming part of algorithm A is the great number of divisions to be executed in pass 2. Fortunately, due to a proposal of E. Elbrächter, we can get rid of them using multiplications instead. This idea is similar to that used in the well known Quadratic Sieve for factoring large integers [5, 12].

We now state the improved version of our algorithm, which is even simpler.

Algorithm B: The multiplicative sieve

Input: $N, M \in \mathbf{N}, I = [N, N+M], M < 2N(N-1)$.
Output: Carmichael number in I.
$P_0 := \frac{1}{4}(1 + \sqrt{8N + 8M + 1})$;
$\mathbf{P}$:= set of odd primes below P_0;
for $i := 0$ to M do $a_i := 1$;
for $p \in \mathbf{P}$ do
begin
  $i(p) := (p - N) \bmod p(p-1)$; (* Here $0 \le i(p) < p(p-1)$ *)
  while $i(p) \le M$ do
  begin
    $a_{i(p)} := a_{i(p)} * p$;
    $i(p) := i(p) + p(p-1)$
  end
end;
for $i := 0$ to M do if $a_i = N + i$ then writeln$(N + i)$;

The correctness of algorithm B depends on

Theorem 6: *After the execution of algorithm B we have* $a_i = \prod_p p$, *where* p *runs through the primes* $p \in \mathbf{P}$ *such that* $p|N+i$ *and* $p-1|N+i-1$.

Proof: Assume $p|a_i$, where $p \in \mathbf{P}$. Then i has the form $i = i(p) + kp(p-1)$ for some k, where $p - N = qp(p-1) + i(p)$ and $0 \leq i(p) < p(p-1)$. Therefore,

$$\begin{aligned} N + i &= N + i(p) + kp(p-1) = N + (p - N - qp(p-1)) + kp(p-1) \\ &= p + p(p-1)(k-q) \equiv p \bmod p(p-1). \end{aligned}$$

Hence $N + i \equiv 0(p)$ and $N + i \equiv 1(p-1)$ for each $p|a_i$.

Conversely, let $p \in \mathbf{P}$ such that $N + i \equiv 0(p)$ and $N + i \equiv 1(p-1)$. Then $N+i \equiv p \bmod p(p-1)$. This leads to $i \equiv p-N \bmod p(p-1)$. Thus $i = i(p)+kp(p-1)$ for some nonnegative integer k and therefore $p|a_i$, q.e.d.

As to the correctness of algorithm B we have to show that $a_i = N + i$ if, and only if, $N + i$ is a Carmichael number in I. Assume $a_i = N + i$. Then Theorem 6 gives

$$N + i = \prod_{\substack{p|N+i \\ p-1|N+i-1}} p,$$

where $p \leq P_0$. Hence, $N + i$ is odd, squarefree, and $p - 1|N + i - 1$ for each $p|N + i$. As in the verification of algorithm A it follows that $N + i$ is composite and consequently is a Carmichael number.

To prove the other direction assume that $N + i$ is a Carmichael number and $0 \leq i \leq M$. If $p|N+i$ then $p \in \mathbf{P}$. Moreover, $N+i \equiv p \bmod p(p-1)$ which implies $i = i(p) + kp(p-1)$. Then a_i contains the factor p. Clearly, these are the only primes occurring in a_i and this concludes the proof.

To use algorithm B effectively it is advisable to apply segmentation. As this extension is straightforward we shall not dwell on details here. We implemented the segmented version of algorithm B on a SIEMENS 7.590 with $M = 10^5 - 1$, $I = \cup_{j=j_0}^{j_1} I_j$ and $I_j = [N_j, N_j + M]$, where $N_j = j(M+1)$. The computation of the Carmichael numbers up to 10^{10} then is a matter of few minutes.

3. Vectorization

From the description of algorithm B it is appearant that the method is well suited for vectorization and we wrote a program for the VP100 located at the University of Kaiserslautern. This machine has a very efficient F77 compiler. It allows integer arithmetic for numbers up to 2^{56}. It is, however, necessary to be careful since integers of length 8 byte have to be declared as real*8. We were ensured by the system administrator that all arithmetical operations (and especially div and mod) are exact, as long as all variables are below the limit 2^{56}.

A naive implementation of algorithm B on the VP100 is not very efficient. In order to take full advantage of the vectorization several modifications were necessary. One of the most important is the different handling of small and large primes of **P** in the while loop of algorithm B. A straightforward FORTRAN simulation would be

```
do 100,j = i(p), M, p(p - 1)
  a(j) = a(j) * p
100 continue
```

This loop is fully vectorized but for large p only few multiplications are performed. Indeed, the number of multiplications is approximately Mp^{-2} for the version of algorithm B as above, and approximately $(j_1 - j_0)Mp^{-2}$ multiplications for the segmented version. If for instance $M = 10^6 - 1$ this means at most one multiplication for the unsegmented algorithm for primes $p \geq 10^3$.

We therefore decided to divide the primes in two classes. The first class consisted of primes $p \leq 10^3$ and the while loop was executed as above. The other primes below P_0 constitute the second class. These have at most one "hit" in an interval $I_j = [N_j, N_j + M]$. At the start of the program the list of primes is scanned to select those primes having a hit in $\cup I_j$. This is done by simply checking whether $i(p)$ has the appropriate value.

Another modification is possible for the least primes, say for the primes $p \leq 17$. The algorithm requires the computation of $i(p)$ such that $j(M + 1) + i(p) \equiv p \bmod p(p - 1)$. Now if p is small and $M + 1 \equiv 0 \bmod p(p - 1)$ we have $i(p) = p$, which is independent of j. Thus p "hits" in each interval I_j at the same set of indices $i(p)$. Hence we choose $M + 1$ to be a multiple of $p(p - 1)$. Then all multiplications of the form $b_{i(p)} := b_{i(p)} * p$ are done in an auxiliary array b and stored. Then only the assignment $a_i := b_i$ is necessary when initializing the array

a. This kind of "small prime variation" (also suggested by E. Elbrächter) also has its counterpart in the Quadratic Sieve for factoring integers [12].

Altogether, our modifications resulted in a tremendous speed-up of the original algorithm. Our latest version is able to sieve $2 \cdot 10^5$ consecutive intervals I_j of the form $I_j = [j(M+1), j(M+1)+M]$ with $M = 4084080 - 1$ in about 45 minutes. The sieving process for an interval of length 10^{12} requires approximately one hour. The running time is nearly independent of the value of N.

4. Results

We used the algorithm described in the last section to compute the Carmichael numbers up to 10^{14}. Let $C(x)$ denote the number of Carmichael numbers less than or equal to x. As mentionend in the introduction the order of magnitude of $C(x)$ is unknown. Based on some plausible hypotheses, Pomerance et al. [13] suggested a growth rate

$$C(x) > x \cdot \exp\bigl(-(2+\varepsilon)\log x \log_3 x / \log_2 x\bigr), \; x \geq x_0(\varepsilon),$$

for each $\varepsilon > 0$. Later, Pomerance [10] refined this to

$$C(x) \sim x \cdot \exp\left(\tfrac{\log x}{\log_2 x}\left(\log_3 x + \log_4 x + \tfrac{\log_4 x - 1}{\log_3 x} + O((\tfrac{\log_4 x}{\log_3 x})^2)\right)\right). \qquad (3)$$

Jäschke [6] gives the approximation

$$C(x) \approx x \cdot \exp\left(-\tfrac{\log x}{\log_2 x}\left(\log_3 x + \log_4 x + \tfrac{\log_4 x - 1}{\log_3 x} + g(x)\right)\right),$$

where

$$g(x) = \frac{7.1287 \log_4^2 x - .2289 \log_4 x + 2.0161}{\log_3^2 x},$$

but this function obviously only was designed to fit the data.

The following table records some values of $C(x)$ found with our algorithm.

Table 1: Distribution of $C(x)$

x	$C(x)$
10^9	646
10^{10}	1547
10^{11}	3605
10^{12}	8241
10^{13}	19279
10^{14}	44706

We note that (3) does not give a reasonable approximation to $C(x)$ for $x \leq 10^{14}$. This is due to the fact that the $\log_4 x$ terms grow to slow to give a significant contribution for our small valuies of x. To obtain better agreement it would be necessary to extend the computations well beyond 10^{20} or even 10^{30} which is out of reach at present.

There are several other interesting quantities associated with Carmichael numbers and we now discuss some of them. By Theorem 2 each Carmichael number is squarefree and we let $C_k(x)$ denote the number of Carmichael numbers up to x having exactly k prime factors. Of course, $C_k(x) = 0$ for $k = 1, 2$ and all x. The following two tables give the values of $C_k(x)$, $C_k(c)/C(x)$ and $\pi_k(x)/x$ for $x = 10^{12}$ and $x = 10^{14}$. Here $\pi_k(x)$ counts the squarefree integers $\leq x$ with exactly k prime factors. By Landau's theorem [7], $\pi_k(x)/x$ is asymptotically Poisson distributed with parameter $\log\log x$, i.e.

$$\frac{\pi_k(x)}{x} \sim e^{-\lambda}\frac{\lambda^{k-1}}{(k-1)!}, \ \lambda = \log\log x,$$

for fixed k and $x \to \infty$.

Table 2: Distribution of $C_k(x)$ for $x = 10^{12}, \log\log x = 3.3189$.

k	$C_k(x)$	$C_k(x)/C(x)$	$e^{-\lambda}\lambda^{k-1}/(k-1)!$
3	1000	.1213	.1993
4	2102	.2551	.2205
5	3156	.3830	.1830
6	1714	.2080	.1215
7	262	.0318	.0672
8	7	.0008	.0132

Table 3: Distribution of $C_k(x)$ for $x = 10^{14}, \log\log x = 3.4731$.

k	$C_k(x)$	$C_k(x)/C(x)$	$e^{-\lambda}\lambda^{k-1}/(k-1)!$
3	3284	.0735	.1871
4	6042	.1351	.2166
5	14938	.3341	.1881
6	14401	.3221	.1306
7	5359	.1199	.0756
8	655	.0147	.0375
9	27	.0006	.0163

A naive guess would be that $C_k(x)/C(x)$ has a similar distribution, but this is false provided $C(x) \sim F(x)$ as in (3) above. In fact, Pomerance et al. [13] have

shown that

$$C_k(X) \leq x^{1-\frac{1}{k}}, \ x \geq x_0(k).$$

The maximum value k_{max} of $C_k(x)/C(x)$ occurs at $k = 4$ for $x = 25 \cdot 10^9$, while $k_{max} = 5$ for $x = 10^{12}$ and $x = 10^{14}$. However, the tables shows a clear tendency towards higher values of k_{max} as x grows. The gap between $C_5(x)/C(x)$ and $C_6(x)/C(x)$ is much smaller for $x = 10^{14}$ than for $x = 10^{12}$.

The next table concerns the distribution of Carmichael numbers in residue classes modulo m for some small moduli m. In each case the residue class $1 \bmod m$ contains by far the most numbers. This is not surprising, taking into account the action of Theorem 2. Let c be a Carmichael number, q prime and $q|c$. If $q \equiv 1(m)$ then also $c \equiv 1(m)$. This fact can be exploited in a quantitative way as follows. We assume that m is an odd prime, $(c, m) = 1$ and $c = q_1 \cdots q_t$, where $t \geq 3$ and the q_i are primes. Let $q_i \equiv a_i(m)$. Our basic assumption is that the q_i are uniformly distributed in the residue classes $1, \ldots, m-1$ so that each a_i occurs with the same probability $(m-1)^{-1}$. If some $a_i = 1$ then $c \equiv 1(m)$; otherwise $c \bmod m$ may take any value $\neq 0$.

If $p_t(m, a)$ denotes the probability that $c \equiv a \bmod m$, we therefore expect

$$p_t(m, 1) = 1 - \left(1 - \frac{1}{m-1}\right)^{t+1},$$

$$p_t(m, a) = \frac{1}{m-1}\left(1 - \frac{1}{m-1}\right)^t, \ 2 \leq a \leq m-1.$$

Table 4 given below shows some correlation between conjectured and empirical data. However, the deviations occurring cast some doubt on the simplified reasoning above.

Table 4: Distribution of Carmichael numbers in residue classes mod 11. The numbers $p_t(m, 1)$ and $p_t(m, a)$ are as above, $\delta_t(m, 1, x) = C_t(m, 1, x)/C_t(m, x)$, where $C_t(m, x)$ denotes the Carmichael numbers $\leq x$ not divisible by m with exactly t prime factors, and $C_t(m, a, x)$ the number of these which are $\equiv a \bmod m$. Finally, $\bar{\delta}_t(m, a, x)$ denotes the arithmetical mean of $\delta_t(m, a, x)$ for $2 \leq a \leq m-1$.

t	3	4	5	6	7	8	9
$p_t(11, 1)$	.3439	.4095	.4686	.5217	.5695	.6126	.6513
$\delta_1(11, 1, x)$	.3360	.4057	.4396	.4629	.4983	.5754	.8333
$p_t(11, a)$	.0729	.0656	.0590	.0531	.0478	.0430	.0387
$\bar{\delta}_t((11, a, x)$	.0738	.0660	.0623	.0597	.0557	.0472	.0185

One should note however, that the above heuristic argument holds only for sufficiently large prime m. For example, if $m = 3$ then necessarily $p_t(3,2) = 0$ if t is even. Also if m is composite the argument has to replaced by a more elaborate reasoning. This is due to the fact that there are residue classes $1 \neq a \bmod m$ such that $d = (a-1, m) > 1$. Consequently such classes should contain more Carmichael numbers. Typical in this respect is the case $m = 12$, see the next table.

Table 5: Distribution of Carmichael numbers $\leq 10^{14}$ in residue classes mod 12. The figures give the number of Carmichaels in each residue class having exactly t prime factors.

t	3	4	5	6	7	8	9
1 mod 12	2890	6022	14502	14335	5330	654	27
3	0	0	2	0	0	0	0
5	122	0	103	0	3	0	0
7	267	0	269	0	11	0	0
9	1	20	62	66	15	1	0
11	4	0	0	0	0	0	0

We now consider the values of $\lambda(c)$, if c is a Carmichael number. Since $\lambda(c)|c-1$ we define an integer $\kappa(c)$ by $\kappa(c) = \frac{c-1}{\lambda(c)}$. The values of κ are distributed irregularly. The least value for the range $x \leq 10^{14}$ occurs at the sixth Carmichael number $c = 6601$ where $\kappa(c) = 5$. But small values for κ continue to appear even for large c. For example $\kappa(42018333841) = 18$ and others are given in table 6.

Table 6: Carmichael numbers with small values of $\kappa(c)$.

c	Factorization	$\kappa(c)$
561	$3 \cdot 11 \cdot 17$	7
1105	$5 \cdot 13 \cdot 17$	23
2465	$5 \cdot 17 \cdot 29$	22
6601	$7 \cdot 23 \cdot 41$	5
10585	$5 \cdot 29 \cdot 73$	21
11921001	$3 \cdot 29 \cdot 263 \cdot 521$	25
55462177	$17 \cdot 23 \cdot 83 \cdot 1709$	18
8885251441	$11 \cdot 47 \cdot 1109 \cdot 15497$	18
42018333841	$11 \cdot 47 \cdot 1049 \cdot 77477$	18

It is conceivable that

$$\liminf \kappa(c) < \infty,$$

which is of course only significant if there are infinitely many Carmichael numbers.

To get a better picture of the distribution of $\kappa(c)$ define

$$\alpha(c) = \frac{\log \lambda(c)}{\log c}.$$

Then $0 < \alpha(c) < 1$. Figure 1 shows the histogram of the numbers

$$n_k = \frac{\delta}{C(x)} |\{c \leq x | k \leq 100\alpha(c) < k+1\}|$$

for $x = 10^{12}$ and figure 2 the same quantities for $x = 10^{14}$.

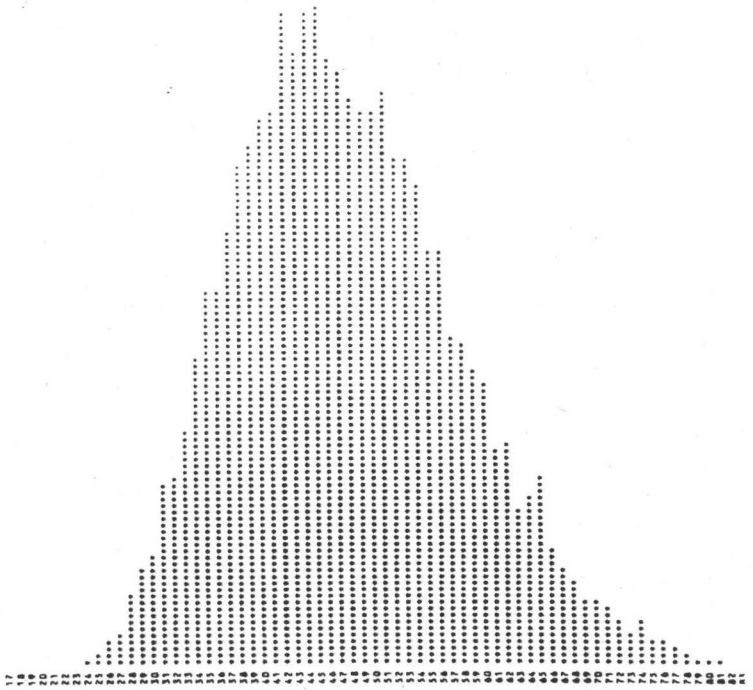

Fig. 1 Empirical distribution of $\alpha(c)$ for $x = 10^{12}$.

The constant δ is chosen so that the maximum value of n_k equals 100. The figures suggest a limiting distribution of the $\alpha(c)$ for $x \to \infty$. For example, from fig. 2 it appears that $\alpha(c)$ is normally distributed with mean 0.43 and standard deviation 0.9 for the Carmichael numbers $\leq 10^{14}$. This behaviour would be in sharp contrast to the normal order of $\alpha(n)$, where n runs over all integers. In fact, as Erdös et al. have shown [4], the normal order of $\alpha(n)$ approaches zero as $n \to \infty$.

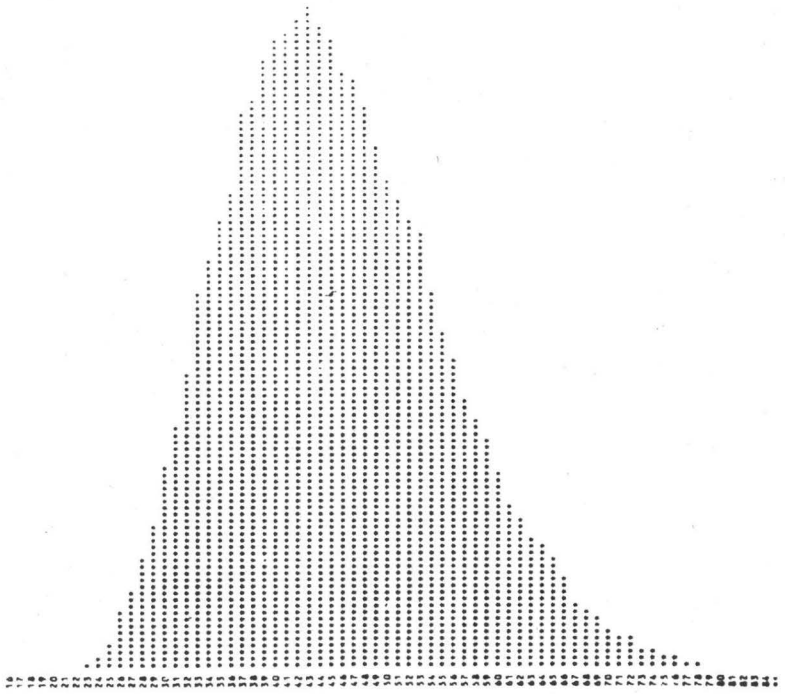

Fig. 2 Empirical distribution of $\alpha(c)$ for $x = 10^{14}$.

Similar statistics were made for the size of the largest prime factor $p(c)$ of c. Here we let

$$\beta(c) = \frac{\log p(c)}{\log c}.$$

Then $0 < \beta(c) < \frac{1}{2}$ by Theorem 5. Figure 3 and 4 show the distribution of the values of

$$m_k = \frac{\delta}{C(x)} |\{c \leq x | k \leq 200\beta(c) < k+1\}|.$$

Here the constant δ is chosen similarly as above. From the figures it appears that the $\beta(c)$ are not distributed arbitrarily but it is not easy to guess the right distribution function (provided there exists one!). The maximum value of n_k occurs at $k = 63$ for $x = 10^{12}$ and at $k = 52$ for $x = 10^{14}$.

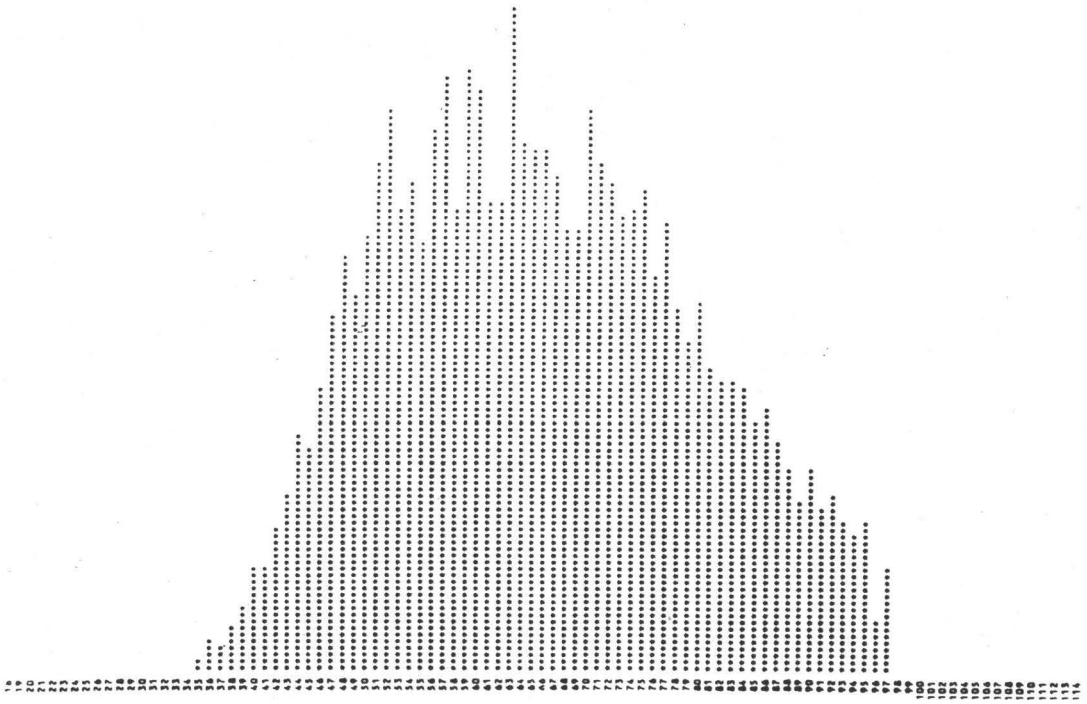

Fig. 3 Empirical distribution of $\beta(c)$ for $x = 10^{12}$.

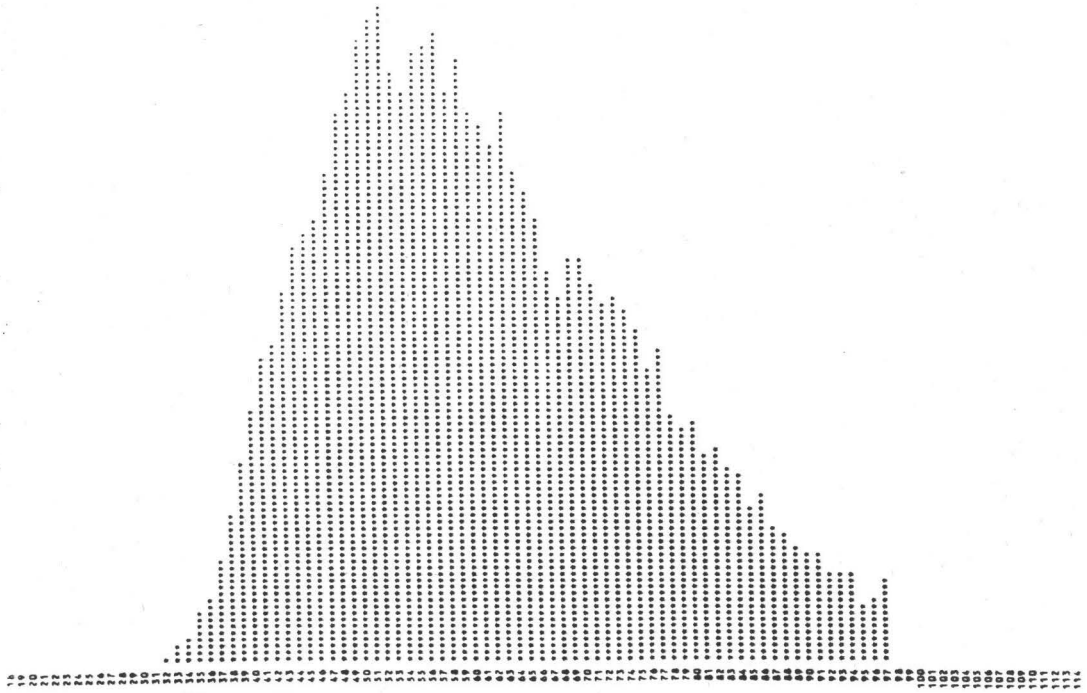

Fig. 4 Empirical distribution of $\beta(c)$ for $x = 10^{14}$.

It seems plausible to conjecture

$$\liminf \beta(c) = 0, \ \limsup \beta(c) = \tfrac{1}{2},$$

provided there are infinitely many Carmichael numbers.

5. The Construction of Carmichael Numbers

In this final section we give a method for the construction of Carmichael numbers. We note that the idea already occurs in the report of Pomerance et al.[13], where it relates to the conjectured lower bound of $C(x)$. Let m be a positive integer and denote by $\rho(m)$ the set of odd primes p such that $p-1|m$ and $(p,m)=1$. Define $P_m = \prod_{p\in\rho(m)} p$ and let c be a divisor of P_m such that $c \equiv 1(m)$ and $c > m+1$. Then c is odd, squarefree and if $p|c$ then $c-1 \equiv 0(m)$, hence $c-1 \equiv 0(p-1)$. It follows that c is a Carmichael number.

This suggests to construct Carmichael numbers as follows. Let m be such that there are many, say t, primes p with $p-1|m$. Then compute all divisors of P_m which are congruent 1 modulo m. Appropriate candidates for m are readily found. They also occur in the primality test of Cohen and Lenstra [3]. The computation of the 2^t divisors of P_m is easily done by using the Gray code for the generation of all subsets of a t-set [8, 9]. In this way we obtained the following tables. $N(m)$ denotes the number of Carmichael numbers produced by m and $C(\nu,m)$ is the number of Carmichael numbers $c|P_m$ with exactly ν prime factors. The tables show a remarkable symmetry with respect to ν of the values of $C(\nu,m)$. It reflects the symmetry of the binomial distribution and is due to the fact that the divisors of P_m are distributed uniformly modulo m. Compare the remarks of Pomerance et al. ([13], p. 1016).

Table 7: Carmichael numbers c produced with $m = 10080, |\rho(m)| = 27, N(m) = 58058$.

ν	3	4	5	6	7	8	9	10
$C(\nu,m)$	2	10	33	131	390	927	2025	3621
ν	11	12	13	14	15	16	17	18
$C(\nu,m)$	5655	7567	8711	8673	7529	5630	3639	2029
ν	19	20	21	22	23	24		
$C(\nu,m)$	956	366	128	31	4	1		

Table 8: Carmichael numbers c produced with $m = 15120, |\rho(m)| = 32, N(m) = 1244091$.

ν	3	4	5	6	7	8	9
$C(\nu, m)$	0	13	63	283	987	3046	8167
ν	10	11	12	13	14	15	16
$C(\nu, m)$	18710	37390	65419	100426	136569	163934	173884
ν	17	18	19	20	21	22	23
$C(\nu, m)$	163744	136654	100571	65419	37512	18651	8172
ν	24	25	26	27	28	29	30
$C(\nu, m)$	3093	1012	288	66	15	1	1

For the readers amusement we finally present a Carmichael number with two hundred prime factors:

853773001845756661416935844996071755186032747369165
041865448862389417978506841669205674174817956938439
468408773178478243859331031723061589165564712220825
573313639584057660487977692665607321786500446687058
674705532333363539130381861268187906932443615357004
384593761503862379573051830054394381154446507667100
206839140952242739222188734933352448809339460182456
571528355024675755053009701926046494688483280787954
947764874440019843606572070630848945301019045973266
279992043350689568091273818743822131150088613985634
936984262033153925948005385077999051636105474468971
303983684815388084209037379211396844998858005083408
371111461909964726241281157138392057414969927294653
741500603215752685397882591747774203309444965811525
1975430315041292641 =

19·23·29·31·37·41·43·53·61·67·71·73·79·89·97·103·109·113·127·131·137·157·181·199·211·239·241·271·281·307·313·331·337·353·379·397·409·421·433·443·463·521·541·547·613·617·631·661·673·757·859·881·911·919·937·953·991·1009·1021·1093·1123·1171·1249·1321·1327·1361·1429·1531·1871·1873·2003·2017·2081·2143·2311·2341·2377·2381·2521·2731·2857·2861·2971·3061·3121·3169·3361·3433·3511·3571·

3673·3697·4159·4421·4591·4621·5237·5281·6007·6121·6427·6553·6733·7393·7481·
7561·7723·8009·8161·8191·8317·8581·8737·9181·9241·9283·9521·9829·10711·12241·
12377·12853·13729·14281·14561·15121·15913·16381·16633·16831·17137·17681·
18481·19891·20021·20593·21841·22441·23563·23761·23869·24481·24571·25741·
26209·27847·28081·29173·29921·30241·30941·34273·36037·36721·38611·39313·
42841·43759·46411·47521·48049·51481·52361·53857·55441·59671·63649·65521·
66529·70687·72073·72931·74257·78541·79561·87517·92821·96097·97241·100981·
102103·108109·110881·116689·117811·120121·123553·128521·131041·145861·
148513·157081·161569·167077·180181·185641·196561·201961·209441·216217

It is a pleasure for us to acknowledge the help of several people. As already mentioned, Eike Elbrächter contributed some significant ideas to the algorithms in section 2 and 3. Much thanks are also due to Dr. T. Pool for his excellent introduction to the VP100 system and for his advice how to use it. We also extend our gratitude to Dr. M. Bürkle of the Rechenzentrum for the generous provision of CPU time on the VP100.

Bibliography

[1] Bays, C., Hudson, R.H., The Segmented Sieve of Eratosthenes and Primes in Arithmetic Progressions to 10^{12}, BIT **17**, 121-129(1977).

[2] Carmichael, R.D., On Composite Numbers P Which Satisfy the Fermat Congruence $a^{P-1} \equiv 1 \bmod P$, Amer. Math. Monthly **19**, 22-27(1912).

[3] Cohen, H., Lenstra, H.W.Jr., Primality Testing and Jacobi Sums, Math. Comp **42**, 297-330(1984).

[4] Erdös, P., Pomerance, C., Schmutz, E., Carmichael's lambda function, Acta Arith. **58**, 363-385(1991).

[5] Gerver, J.L., Factoring Large Integers With a Quadratic Sieve, Math. Comp. **41**, 287-294(1983).

[6] Jäschke, G., The Carmichael Numbers to 10^{12}, Math. Comp. **55**, 383-389(1990).

[7] Landau, E., Handbuch der Lehre von der Verteilung der Primzahlen, Leipzig: Teubner 1909.

[8] Lüneburg, H., Tools and Fundamental Constructions of Combinatorial Mathematics, Mannheim 1989.

[9] Nijenhius, A., Wilf, H.S., Combinatorial Algorithms, New York 1978.

[10] Pomerance, C., On the Distribution of Pseudoprimes, Math. Comp. **37**, 587-593(1981).

[11] Pomerance, C., A New Lower Bound for the Pseudoprime Counting Function, Ill. J. Math. **26**, 4-9(1982).

[12] Pomerance, C., Factoring, in: Cryptology and Computational Number Theory, Proc. Symp. Applied Math., Vol. 42, 1990, 27-47.

[13] Pomerance, C., Selfridge, J.L., Wagstaff, S.S., Jr., The Pseudoprimes to $25 \cdot 10^9$, Math. Comp. **35**, 1003-1026(1980).

[14] Swift, J.D., Table of Carmichael Numbers to 10^9, Math. Comp. **29**, 338-339(1975).