COUNTING CARMICHAEL NUMBERS WITH SMALL SEEDS

ZHENXIAO ZHANG

Abstract. Let $A_s$ be the product of the first $s$ primes, let $P_s$ be the set of primes $p$ for which $p-1$ divides $A_s$ but $p$ does not divide $A_s$, and let $C_s$ be the set of Carmichael numbers $n$ such that $n$ is composed entirely of the primes in $P_s$ and such that $A_s$ divides $n-1$. Erdős argued that, for any $\varepsilon > 0$ and all sufficiently large $x$ (depending on the choice of $\varepsilon$), the set $C_s$ contains more than $x^{1-\varepsilon}$ Carmichael numbers $\leq x$, where $s$ is the largest number such that the $s$th prime is less than $\ln x^{\varepsilon/4}$. Based on Erdős’s original heuristic, though with certain modification, Alford, Granville, and Pomerance proved that there are more than $x^{3/7}$ Carmichael numbers up to $x$, once $x$ is sufficiently large.

The main purpose of this paper is to give numerical evidence to support the following conjecture which shows that $|C_s|$ grows rapidly on $s$: $|C_s| = 2^{2^s(1-\varepsilon)}$ with $\lim_{s \to \infty} \varepsilon = 0$, or, equivalently, $|C_s| = A_s^{2^{s(1-\varepsilon')}}$ with $\lim_{s \to \infty} \varepsilon' = 0$.

We describe a procedure to compute exact values of $|C_s|$ for small $s$. In particular, we find that $|C_6| = 8,281,366,855,879,527$ with $\varepsilon = 0.36393\ldots$ and that $|C_{10}| = 21,823,464,288,660,480,291,170,614,377,509,316$ with $\varepsilon = 0.31662\ldots$. The entire calculation for computing $|C_s|$ for $s \leq 10$ took about 1,500 hours on a PC Pentium Dual E2180/2.0GHz with 1.99 GB memory and 36 GB disk space.

1. Introduction

Let $b_i$ be the $i$th prime. Let $s \geq 1$ and let $A_s = \prod_{i=1}^{s} b_i$ be the product of the first $s$ primes. It is easy to see that (as Erdős knew)

\begin{equation}
A_s < e^{2b_s}.
\end{equation}

Define sets

\begin{equation}
P_s = \{ \text{prime } p : p > b_s, \ p - 1 | A_s \},
\end{equation}

\begin{equation}
N_s = \{ n > 1 : n \text{ is square free and composed entirely of the primes in } P_s \},
\end{equation}

\begin{equation}
C_s = \{ n \in N_s : A_s | n - 1, n - 1 \neq A_s \}.
\end{equation}

By Korselt’s criterion (see also [3, Section 3.4.2]), every number $n \in C_s$ is Carmichael [2]. Since the sets $P_s, N_s$, and $C_s$ are determined by the first $s$ primes, we say that these sets are generated by the (square-free) (prime) seeds $b_1, \ldots, b_s$.

Received by the editor October 19, 2009 and, in revised form, October 28, 2009.

2010 Mathematics Subject Classification. Primary 11Y16, 11Y35; Secondary 11Y11.

Key words and phrases. Carmichael numbers (with small seeds), Korselt’s criterion, heuristics of Erdős-AGP concerning Erdős’s construction of Carmichael numbers, product of the first $s$ primes, algorithms.

The author was supported by the NSF of China Grant 10071001.

©2010 American Mathematical Society
Reverts to public domain 28 years from publication.
ZHENXIANG ZHANG

Erdős [4] argued that, for any \( \varepsilon > 0 \) and all sufficiently large \( x \) (depending on the choice of \( \varepsilon \)), the set \( C_s \) contains more than \( x^{1-\varepsilon} \) Carmichael numbers \( \leq x \), where \( s \) is the largest number such that \( b_s < \ln x^{\varepsilon/4} \). In short, Erdős [4] made the following Conjecture 1.

**Conjecture 1 (Erdős).** There are \( x^{1-o(1)} \) Carmichael numbers up to \( x \).

Based on Erdős’s original heuristic [4], though with certain modification, Alford, Granville, and Pomerance [1] proved the following Theorems 1 and 2.

**Theorem 1 (Alford, Granville, and Pomerance).** There are more than \( x^{2/7} \) Carmichael numbers up to \( x \), once \( x \) is sufficiently large.

**Theorem 2 (Alford, Granville, and Pomerance).** Fix \( \varepsilon > 0 \). Assume that, for sufficiently larger \( x \), the arithmetic progression \( 1 \pmod{d} \) contains more than \( x/(2d \ln x) \) primes up to \( x \) provided \( d < x^{1-\varepsilon} \). Then there are more than \( x^{1-2\varepsilon} \) Carmichael numbers up to \( x \), once \( x \) is sufficiently large.

Note that the counts of the number of Carmichael numbers in either Conjecture 1 or Theorems 1 and 2 are functions which grow slowly on \( x \). For \( x = 10^n \) for \( n \) up to 21 (which is as far as has been computed [7]), there are fewer than \( x^{0.348} \) Carmichael numbers up to \( x \).

The main purpose of this paper is to give numerical evidence to support the following Conjecture 2, which shows that \( |C_s| \) grows rapidly on \( s \).

**Conjecture 2.** We have

\[
|C_s| = 2^{2^s(1-\varepsilon)}
\]

with \( \lim_{s \to \infty} \varepsilon = 0 \), or, equivalently,

\[
|C_s| = A_s^{2^{s(1-\varepsilon')}}
\]

with \( \lim_{s \to \infty} \varepsilon' = 0 \).

In Section 2, we first briefly state reasons for making Conjecture 2 which are essentially based on the heuristics of Erdős, Alford, Granville, and Pomerance concerning Erdős’s construction of Carmichael numbers. Then we describe a procedure for finding \( |C_s| \) for small \( s \) and tabulate \( |C_s| \) and relative values for \( 3 \leq s \leq 10 \). In particular, we have \( |C_9| = 8,281,366,855,879,527 \) with \( \varepsilon = 0.36393 \ldots \) and \( |C_10| = 21,823,464,288,660,480,291,170,614,377,509,316 \) with \( \varepsilon = 0.31662 \ldots \). The entire calculation for \( |C_s| \) for \( s \leq 10 \) took about 1,500 hours on a PC Pentium Dual E2180/2.0GHz with 1.99 GB memory and 36 GB disk space.

**Remark 1.1.** Alford (see [5]) took \( L = 2^6 \cdot 3^3 \cdot 5^2 \cdot 7^2 \cdot 11 \), determined 155 primes \( p \) for which \( p-1 \) divides \( L \), and then established that there are at least \( 2^{128} - 1 \) Carmichael numbers made up from them. However, Alford did not express the number of Carmichael numbers as a function of \( L \). Granville [5] mentioned: “It can be shown that if \( L = A_s \) for some sufficiently large \( s \), then we can obtain more than \( 2 \ln^2 L \) primes in \( \mathcal{P}_s \), and so we’d expect more than

\[
L^{\ln^2 L}
\]

Carmichael numbers in \( C_s \).” The estimate (1.7) seems to be the only estimate for \( |C_s| \) in the literature, which grows much more slowly than that in Conjecture 2.
2. Evaluating $|C_s|$ 

Since the probability of a number $\leq m$ to be prime is greater than $\frac{1}{\ln m}$ and since $A_s$ has $2^{s-1}$ even divisors, it is reasonable to conjecture that

\begin{equation}
|P_s| = 2^{s(1-o(1))}.
\end{equation}

Given $s \geq 3$, let $Z_{A_s} = \{0, 1, 2, \ldots, A_s - 1\}$ and let

\begin{equation}
Z_{A_s}^* = \{r \in Z_{A_s} : \gcd(A_s, r) = 1\} = \{1 = u_1 < u_2 < \ldots < u_{\varphi(A_s)}\},
\end{equation}

where $\varphi(\cdot)$ is the Euler function. Define the set

$$\mathcal{R}_s = \{r \in Z_{A_s} : r \equiv n \mod A_s \text{ for some } n \in \mathcal{N}_s\}.$$ 

Then $\mathcal{R}_s \subseteq Z_{A_s}^*$ and $|\mathcal{R}_s| \leq \varphi(A_s)$. For $r \in Z_{A_s}^*$, define the function

$$f_s(r) = \#\{n \in \mathcal{N}_s : n \equiv r \mod A_s\}.$$

Then we have

\begin{equation}
|C_s| = \begin{cases} 
  f_s(1) - 1, & \text{if } A_s + 1 \in P_s; \\
  f_s(1), & \text{otherwise.}
\end{cases}
\end{equation}

Let

\begin{equation}
a_s = \frac{|\mathcal{N}_s|}{\varphi(A_s)} = \frac{2^{|P_s|} - 1}{\varphi(A_s)},
\end{equation}

$$g_{s,1} = \max\{f_s(r) : r \in Z_{A_s}^*\}, \quad \text{and} \quad g_{s,2} = \min\{f_s(r) : r \in Z_{A_s}^*\}.$$ 

Let $\beta_s$ be such that

\begin{equation}
g_{s,1} - g_{s,2} = a_s^{\beta_s}.
\end{equation}

Numerical evidence (see Table 1) suggests that

\begin{equation}
\beta_s < 0.6 \quad \text{for } s \geq 8,
\end{equation}

which implies that

$$g_{s,1} - g_{s,2} = o(a_s) \quad \text{and} \quad \lim_{s \to \infty} g_{s,2}/g_{s,1} = \lim_{s \to \infty} |C_s|/a_s = 1.$$

Note that (2.6) gives an explicit and extended version of Erdős’s argument \[4\] that members of the set $\mathcal{N}_s$ are roughly equi-distributed mod $A_s$.

Combining (2.3), (1.1), (2.1), and (2.6), we have Conjecture 2. Based on (2.3), we use the following procedure to compute $|C_s|$ for small $s$.

**PROCEDURE 1.** Finding $|C_s|$:

{input $s \geq 3$, output $g_{s,1}$, $g_{s,2}$, and $|C_s|$, etc.}

**BEGIN** Compute $A_s$ and $\varphi(A_s)$;

Determine the set $P_s = \{p_1 < p_2 < \ldots < p_m\}$;

$i \leftarrow 0$;

For $r := 1$ To $A_s - 1$ Do

begin If $\gcd(A_s, r) = 1$ Then Begin $i \leftarrow i + 1; u_i \leftarrow r; h_r \leftarrow i$ End end;

For $i := 1$ To $\varphi(A_s)$ Do $H(i) \leftarrow 0$;

$i \leftarrow 1; t \leftarrow 1; H(h_{p_1}) \leftarrow 1$;

Repeat $i \leftarrow i + 1; p \leftarrow p_i \mod A_s; H_0 \leftarrow H$;

For $j := 1$ To $\varphi(A_s)$ Do

begin If $H_0(j) > 0$ Then
Begin $r \leftarrow p \cdot u_j \mod A_s$;
If $H(h_s) = 0$ Then $t \leftarrow t + 1$;
$H(h_s) \leftarrow H(h_r) + H_0(j)$
End

end;
If $H(h_p) = 0$ Then $t \leftarrow t + 1$;
$H(h_p) \leftarrow H(h_p) + 1$;
$g_1 \leftarrow \max \{H(j) : 1 \leq j \leq \varphi(A_s)\}$;
If $t < \varphi(A_s)$ Then $g_2 \leftarrow 0$ Else $g_2 \leftarrow \min \{H(j) : 1 \leq j \leq \varphi(A_s)\}$;
Output($i, p_i, g_1, g_2, H(1)$)
Until $i = |P_s|$;
$g_{s,1} \leftarrow g_1$; $g_{s,2} \leftarrow g_2$; $f_s(1) \leftarrow H(1)$;
Determine $|C_s|$ by \[\text{Box}\]

END.

The Delphi-Pascal program (with multi-precision package partially written in Assembly language) ran about 1,500 hours on a PC Pentium Dual E2180/2.0GHz (with 1.99 GB memory and 36 GB disk space) to get $|C_s|$ and relative values for $3 \leq s \leq 10$ tabulated in Table 1.

**Table 1.** $|C_s|$ and relative values for $3 \leq s \leq 10$

| $s$ | $A_s$ | $\varphi(A_s)$ | $|P_s|$ | $|R_s|$ | $a_s$ | $g_{s,1}$ | $g_{s,2}$ | $f_s(1)$ | $|C_s|$ |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 30 | 8 | 3 | 4 | 0 | 2 | 0 | 1 | 0 |
| 4 | 210 | 48 | 5 | 16 | 0 | 2 | 0 | 1 | 0 |
| 5 | 2310 | 480 | 9 | 192 | 1 | 6 | 0 | 3 | 2 |

| $s$ | $A_s$ | $\varphi(A_s)$ | $|R_s|$ | $|P_s|$ | $a_s$ |
|---|---|---|---|---|---|
| 6 | 30030 | 5760 | 17 | 22 |
| 7 | 510510 | 92160 | 28 | 2912 |
| 8 | 9699690 | 1658880 | 50 | 678710881 |
| 9 | 2239092870 | 36495360 | 78 | 8281366587523928 |
| 10 | 6469693230 | 1021870080 | 144 | 21823464288660475450593208749832817 |

<table>
<thead>
<tr>
<th>$s$</th>
<th>$g_{s,2}$</th>
<th>$g_{s,1} - g_{s,2}$</th>
<th>$\beta_s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>9</td>
<td>30</td>
<td>1.10033…</td>
</tr>
<tr>
<td>7</td>
<td>2728</td>
<td>381</td>
<td>0.74502…</td>
</tr>
<tr>
<td>8</td>
<td>678670201</td>
<td>95809</td>
<td>0.56403…</td>
</tr>
<tr>
<td>9</td>
<td>8281366006950486</td>
<td>921747209</td>
<td>0.56317…</td>
</tr>
<tr>
<td>10</td>
<td>21823464288660451215882006081060134</td>
<td>36359681036872185925</td>
<td>0.56963…</td>
</tr>
</tbody>
</table>

| $s$ | $g_{s,2}/g_{s,1}$ | $f_s(1) = |C_s|$ | $\epsilon$ | $\epsilon'$ |
|---|---|---|---|---|
| 6 | 0.23076923… | 30 | 0.61753… | 1.26665… |
| 7 | 0.87745255… | 2896 | 0.49663… | 1.10306… |
| 8 | 0.99985884… | 678687138 | 0.39066… | 0.95774… |
| 9 | 0.99999988… | 8281366855879527 | 0.36393… | 0.89654… |
| 10 | 0.9999999… | 2182346428866048029170614377509316 | 0.31662… | 0.81926… |

**Remark 2.1.** For $s \leq 9$, we save the set $\{u_i\}$ (see (2.2)) in an array with each entry 4 bytes, which takes $\varphi(A_9) \cdot 4 = 145,981,440$ bytes of memory, and save the set $\{h_r : 1 \leq r < A_s, h_r = i \text{ if } r = u_i\}$ also in an array with each entry 4 bytes, which takes $(A_9 - 1) \cdot 4 = 892,371,476$ bytes of memory, since $A_9 = 223,092,870$
and \( \varphi(A_9) = 36,495,360 \) are 4-byte (32-bit) LongWords. Since \( 2^{32} < g_{9,1} = 8,281,366,928,697,695 < 2^{63} \), we save functions \( H \) and \( H_0 \) in arrays with each entry 8 bytes, which take \( \varphi(A_9) \cdot 8 \cdot 2 = 583,925,760 \) bytes of memory. In total, for saving these variables and functions, it takes about 1.63 GB of memory which is fit for my PC Pentium Dual E2180/2.0GHz with 1.99 GB of memory. It took only about 0.5 hours on my PC for computing \( |C_s| \) and relative values for \( 3 \leq s \leq 9 \).

**Remark 2.2.** For \( s = 10 \), the computation becomes much harder. Since

\[
A_{10} = 6,469,693,230 > 2^{32} \quad \text{and} \quad \varphi(A_{10}) = 1,021,870,080,
\]

neither the set \( \{h_r \} \) nor the set \( \{u_i \} \) could be fit in the 1.99 GB of memory of my PC. We have to take a new approach for \( s = 10 \) different from that for \( s \leq 9 \). Note that \( A_8 = 9,699,690 \) and \( \varphi(A_8) = 1,658,880 \). Write

\[
Z_{A_8}^s = \{v_1 < v_2 < \ldots < v_{\varphi(A_8)}\}.
\]

For \( r \in Z_{A_8}^s \), define \( h_r^{(8)} = i \) if \( r = v_i \) for some \( 1 \leq i \leq \varphi(A_8) \). Let

\[
\mathcal{R} = \{1 \leq r < A_{10} : \gcd(A_8, r) = 1\} = \{1 = r_1 < r_2 < \ldots < r_{|\mathcal{R}|}\},
\]

which is a set a little larger than \( Z_{A_8}^{10} \) and contains \( 1 \leq r < A_{10} \) with \( 23|r \) or \( 29|r \). Then \( |\mathcal{R}| = \varphi(A_8) \cdot 23 \cdot 29 = 1,106,472,960 \). For \( r \in \mathcal{R} \) define

\[
\xi(r) = \lfloor r/A_8 \rfloor : \varphi(A_8) + h_r^{(8)} \mod A_8.
\]

For \( 1 \leq j \leq |\mathcal{R}| \) define

\[
\eta(j) = \begin{cases} A_8 \cdot \lfloor (j-1)/\varphi(A_8) \rfloor + v_{\varphi(A_8)}, & \text{if } \varphi(A_8)|j, \\ A_8 \cdot \lfloor (j-1)/\varphi(A_8) \rfloor + v_j \mod \varphi(A_8), & \text{otherwise}. \end{cases}
\]

Then for \( r \in \mathcal{R} \) and \( 1 \leq j \leq |\mathcal{R}| \), we have \( \eta(\xi(r)) = r \) and \( \xi(\eta(j)) = j \). Now the function \( \xi(r) \) serves for \( s = 10 \) as \( h_r \) serves for \( s \leq 9 \), and the function \( \eta(j) \) serves for \( s = 10 \) as \( u_i \) serves for \( s \leq 9 \). The differences are that, for \( s = 10 \), both \( \xi(r) \) and \( \eta(j) \) are computed instantly and frequently, and only the sets \( \{v_i\} \) and \( \{h_r^{(8)}\} \) are saved as arrays in memory, which take only

\[
(A_8 - 1) \cdot 4 + \varphi(A_8) \cdot 4 = 45,434,276
\]

bytes of memory. In the “Repeat . . . Until” loop of Procedure 1, the “For \( j := 1 \) To \( \varphi(L) \) Do begin . . . end” sub-loop is replaced by the following code:

**For \( j := 1 \) To \( |\mathcal{R}| \) Do**

**begin**

**If** \((H_0(j) > 0) \quad \text{And } \quad (\gcd(\eta(j), 23 \cdot 29) = 1)\)** **Then**

**Begin**

\[
\begin{aligned}
& r \leftarrow p \cdot \eta(j) \mod A_{10}; \\
& \text{If } H(\xi(r)) = 0 \text{ Then } t \leftarrow t + 1; \\
& H(\xi(r)) \leftarrow H(\xi(r)) + H_0(j)
\end{aligned}
\]

**End**

**end.**

**Remark 2.3.** In any event, the arrays \( H(j) \) and \( H_0(j) \) \((1 \leq j \leq |\mathcal{R}|)\) for \( s = 10 \) could not be saved in the memory of my PC. They are saved in disk files. Since

\[
2^{64} < g_{10,1} = 21,823,464,288,660,487,575,563,042,953,246,059 < 2^{128},
\]

it takes \( |\mathcal{R}| \cdot 2 \cdot 128/8 \approx 36 \) GB disk space to store \( H(j) \) and \( H_0(j) \) for \( 1 \leq j \leq |\mathcal{R}| \). Since \( 2^{63} - 1 = 9,223,372,036,854,775,807 \) is the maximum integer in Delphi 6.0, a multi-precision package is needed for \( s = 10 \).
Acknowledgments

I thank Professors C. Pomerance and A. Granville for valuable comments on the original versions of this paper. Special thanks go to the referee for friendly and helpful comments that improved the presentation of this paper.

References

2. R. D. Carmichael, Note on a new number theory function, Bull. A. M. S. 16 (1910), 232–238. MR1558896

Department of Mathematics, Anhui Normal University, 241000 Wuhu, Anhui, People’s Republic of China

E-mail address: zhangzhx@mail.ah.cn
E-mail address: ahnu_zxx@sina.com
URL: http://www.ahnu.edu.cn/site/math/html/zxx.htm