A METHOD OF TABULATING
THE NUMBER-THEORETIC FUNCTION \( g(k) \)

RENAE SCHEIDLER AND HUGH C. WILLIAMS

Abstract. Let \( g(k) \) be the least integer \( > k + 1 \) such that all prime factors of \( g(k) \) are greater than \( k \). The function \( g(k) \) appears to show quite irregular behavior and is hard to compute. This paper describes a method of computing \( g(k) \), using sieving techniques, and provides a table of values of \( g(k) \) for \( k \leq 140 \).

1. Introduction

In a symposium on computers in number theory held in 1969, Erdös [3] presented a paper consisting of problems he felt might be approachable by computational techniques. One of these was to determine an estimate for \( g(k) \), where \( g(k) \) is the least integer \( > k + 1 \) such that all the prime factors of \( g(k) \) must exceed \( k \). In a subsequent paper, Ecklund, Erdös, and Selfridge [2] provided a table of values of \( g(k) \). This table is complete for \( k \leq 40 \); also, three more entries are present for \( k = 42, 46, \) and \( 52 \). These are all the values of \( g(k) \leq 2500000 \) when \( k \leq 100 \).

Very little seems to be known about the behavior of \( g(k) \). It appears to increase rather rapidly with increasing \( k \), and it is difficult to compute. Thus, it was thought that a larger table of \( g(k) \) might prove to be useful. The purpose of this paper is to discuss a method of computing \( g(k) \) by using sieving techniques, and to provide a complete table of values of \( g(k) \) for all \( k \leq 140 \).

We begin with a brief discussion of the generalized sieving problem. In general, a sieving problem \( P \) defines \( h \) linear congruences

\[
x \equiv r_{i_1}, r_{i_2}, \ldots, r_{i_h} \pmod{m_i} \quad (i = 1, 2, \ldots, h; 1 \leq n_i < m_i),
\]

where the moduli \( m_1, m_2, \ldots, m_h \) are positive integers. It may be assumed that the \( m_i \) are relatively prime in pairs, and that each set of admissible residues \( R_i = \{r_{i_1}, r_{i_2}, \ldots, r_{i_n}\} \) contains distinct, nonnegative integers less than \( m_i \). The solution set \( S(P) \) for \( P \) is defined to be all integers \( x \) that lie within an interval or range specified by \( P \), say \( A \leq x \leq b \), such that

\[
x \pmod{m_i} \in R_i \quad (i = 1, 2, \ldots, h)
\]

and satisfy any additional restrictions placed on \( x \) by \( P \).
It is possible to construct very fast special purpose machines for finding solutions to sieve problems. Recently, Stephens and Williams [5] have described such a device, called OASiS. OASiS will search for values of $x$ satisfying a sieving problem at the rate of 215,000,000 trials per second. It should be pointed out that OASiS is just the most recent in a long series of such machines. For a history of the developments, we refer the reader to [5]. When sieving mechanisms are in use, it is customary to call the sets of admissible residues $R_i$ rings modulo $m_i$ and the process of determining values which should be in the $R_i$ sets loading the rings. The execution of any sieve problem is made up of two phases: (1) loading the rings, (2) searching for the solution. The process of searching for a solution is performed by first producing values in the range satisfying (1.1) and then determining whether these values satisfy any additional restrictions. This latter operation is called filtering.

In the next sections we will show how the problem of determining $g(k)$ can be converted to a sieve problem.

2. The algorithm

We need to determine the minimal number $n > k + 1$ such that no prime $p \leq k$ is a divisor of $\binom{n}{k}$. In order to determine whether or not $\binom{n}{k}$ is divisible by a prime $p$, we first make use of a result which is essentially due to Kummer (see [1, p. 220]).

**Theorem 1.** Let $n = \sum_{i=0}^{t} b_i p^i$ and $k = \sum_{i=0}^{t} a_i p^i$ be the base-$p$ representations of $n$ and $k$, respectively, where $p$ is a prime. Then $p \nmid \binom{n}{k}$ if and only if $b_i \geq a_i$ ($i = 0, 1, \ldots, t$).

**Proof.** Let $e_{i-1} = 0$, and for $i = 0, 1, \ldots, t$ put

$$
e_i = \begin{cases} 1 & \text{if } b_i < a_i + e_{i-1} \\ 0 & \text{if } b_i \geq a_i + e_{i-1} \end{cases}, 
\ c_i = p e_i + b_{i-1} - a_{i-1} - e_{i-1}.$$

Then the $e_i$ are the “carry-overs” when performing the subtraction of $k$ from $n$ in base $p$, and we have

$$n - k = \sum_{i=0}^{t} c_i p^i - p^{t+1} e_t$$

and $0 \leq c_i < p$. Since $n - k > 0$, we must have $e_t = 0$, and we have found that the base-$p$ representation of $n - k$ is given by

$$n - k = \sum_{i=0}^{t} c_i p^i.$$

We can now use the well-known theorem of Legendre on the highest power of $p$ which divides the factorial of an integer to find that

$$p^\alpha \| \binom{n}{k},$$

where $\alpha = \sum_{i=0}^{t} e_i$. It follows that $p \nmid \binom{n}{k}$ if and only if $e_0 = e_1 = \cdots = e_t = 0$, and this occurs if and only if $a_i \leq b_i$ ($i = 0, 1, \ldots, t$). $\square$
In order to make use of this past result, we need to be able to compute the coefficients of $k$ in base $p$. If $k = \sum_{i=0}^{m} a_{i} p^{i}$, where $m$ is such that $p^{m} \leq k < p^{m+1}$, i.e., $m = \lfloor \log_{p} k \rfloor = \lfloor \log k / \log p \rfloor \geq 1$, then the coefficients $a_{i}$ ($i = 0, 1, \ldots, m$) can be easily computed by putting

\[
a_{0} \equiv k \pmod{p} \quad (0 < a_{0} < p), \quad s_{0} = k,
\]
\[
a_{i+1} \equiv s_{i+1} \pmod{p} \quad (0 < a_{i+1} < p), \quad s_{i+1} = \frac{s_{i} - a_{i}}{p}.
\]

Since we would like to convert our condition $p \nmid \binom{n}{k}$ into a sieving problem, we must determine the possible residues of $n$ modulo $p^{m}$. Define the sets $C_{i}$ ($i = 0, 1, \ldots, m$) by

\[C_{i} = \{a_{i}, a_{i+1}, a_{i+2}, \ldots, p-1\}.\]

If $n = \sum_{i=0}^{t} b_{i} p^{i}$, then $t \geq m$, and by Theorem 1, $p \nmid \binom{n}{k}$ if and only if $b_{i} \in C_{i}$ for all $i = 0, 1, \ldots, m$. For two arbitrary sets $S, T$ of integers and a rational number $l$, let

\[S + T = \{s + t | s \in S, t \in T\}, \quad lS = \{ls | s \in S\}, \quad S + l = S + \{l\}, \quad \frac{S}{l} = \frac{1}{l} S.\]

We now define sets

\[B_{0} = C_{0}, \quad B_{i} = B_{i-1} + p^{i} C_{i} \quad (i = 1, 2, \ldots, m).\]

It follows that $B_{i} = \sum_{j=0}^{i} p^{j} C_{j}$.

**Lemma 1.** Let $n' \equiv n \pmod{p^{m}}$ and $0 \leq n' < p^{m}$. Then $n' \in B_{m-1}$ if and only if $b_{i} \in C_{i}$ ($i = 0, 1, \ldots, m-1$).

**Proof.** It is easy to see that if $b_{i} \in C_{i}$ ($i = 0, 1, \ldots, m-1$), then $n' = \sum_{i=0}^{m-1} b_{i} p^{i} \in B_{m-1}$. The rest of the lemma follows from the fact that $p^{i} C_{i} \cap p^{j} C_{j} = \emptyset$ for $i \neq j$. \[\square\]

From Theorem 1 and Lemma 1 it follows that $p \nmid \binom{n}{k}$ if and only if $n' \in B_{m-1}$ and $b_{m} \in C_{m}$. If we write $n = n' + p^{m} y$, then $y = \sum_{i=0}^{m} b_{m+i} p^{i}$; hence, $y \equiv b_{m} \pmod{p}$. Since $n/p^{m} \leq y < n/p^{m+1}$, we have $b_{m} \equiv \lfloor n/p^{m} \rfloor \pmod{p}$. If we let $B_{m-1} = \{r_{1}, r_{2}, \ldots, r_{q}\}$, we obtain the following result.

**Theorem 2.** We have $p \nmid \binom{n}{k}$ if and only if $n \equiv r_{1}, r_{2}, \ldots, r_{q} \pmod{p^{m}}$ and $[n/p^{m}] \pmod{p} \geq a_{m}$.

This gives rise to the following sieving algorithm for determining $g(k)$.

(a) Load the rings. For each prime $p \leq k$:

1. Compute $m = \lceil \log k / \log p \rceil$ and determine $a_{0}, a_{1}, \ldots, a_{m}$ as in (2.1).
2. Find $B_{m-1} = \{r_{1}, r_{2}, \ldots, r_{q}\}$.
3. Load the values of $r_{1}, r_{2}, \ldots, r_{q}$ into the ring of modulus $p^{m}$.
(b) Search for \( g(k) \).

1. Start searching for solutions at \( 2k + 1 \). (In [2] it is shown that \( g(k) \geq 2k + 1 \) for \( k > 4 \).)
2. Once a certain range has been sieved, test each solution candidate \( s \) by the following routine: for each \( p \leq k \) determine that \( \left\lfloor \frac{s}{p^m} \right\rfloor \pmod{p} \geq a_m \).
3. The least value of \( s \) which passes this test is the value of \( g(k) \).

The implementation of this algorithm produced immense sieving times for values of \( g(k) \) for even modest values of \( k \). Fortunately, it is possible to speed up the computation by a factor of approximately \( k \) in the case when \( k + 1 \) is a composite integer. To describe this faster algorithm, assume that \( k + 1 \) is composite, and write \( k + 1 = q^a \alpha \) (where \( p \) is a prime and \( p \not| q \)).

**Lemma 2.** If \( k + 1 \) is composite, then \( k + 1 | g(k) + 1 \).

**Proof.** If \( k + 1 \) is composite, we have \( p < k \) and

\[
\begin{align*}
k &= q^a \alpha - 1 = (q - 1)p^a + p^a - 1 \\
&= (q - 1)p^a + (p - 1)(p^{a-1} + p^{a-2} + \cdots + p + 1);
\end{align*}
\]

hence, \( a_0 = a_1 = \cdots = a_{a-1} = p - 1 \) and \( C_0 = C_1 = \cdots = C_{a-1} = \{p - 1\} \). From this it is easy to show that

\[
B_i = \{p^{i+1} - 1\} \quad (0 \leq i \leq \alpha - 1).
\]

From our earlier results it is easy to deduce that \( p \not| \binom{\alpha}{k} \) only if \( n \pmod{p^a} \in B_{a-1} \); that is, if \( p \not| \binom{\alpha}{k} \) then \( n \equiv -1 \pmod{p^a} \). Since this must be true for all prime divisors of \( k + 1 \), we see that when \( k + 1 \) is composite we must have \( k + 1 | g(k) + 1 \). \( \square \)

Thus, in this case, \( h(k) = (g(k) + 1)/(k + 1) \) is an integer, and in fact, \( N = (n + 1)/(k + 1) \) must be an integer for all possible values \( n \) for \( g(k) \). Hence, we can increase the speed of sieving by a factor of \( k + 1 \) by sieving for \( h(k) \) instead of \( g(k) \). We now explain how this can be done. We first require

**Theorem 3.** Let \( n \) be a possible value for \( g(k) \), where \( k + 1 \) is composite, and let \( N = (n + 1)/(k + 1) \). Let \( p \leq k \) be a prime.

(a) If \( p \not| k + 1 \), then \( N \pmod{p^m} \in Q_m(B_{m-1} + 1) \), where

\[
\tag{2.2}
Q_m(k + 1) \equiv 1 \pmod{p^m}.
\]

(b) If \( p^\alpha \parallel k + 1 \), then we have two cases:

if \( \alpha = m + 1 \), then \( n \equiv -1 \pmod{p^m} \);
if \( \alpha \leq m \), then \( N \pmod{p^m} \in Q_m(B_{m-1} + 1)/p^\alpha \), where

\[
\tag{2.3}
Q_m \frac{k + 1}{p^\alpha} \equiv 1 \pmod{p^m}.
\]

**Proof.** By Lemma 1, we have \( n + 1 \pmod{p^m} \in B_{m-1} + 1 \).

(a) Let \( p \not| k + 1 \) and let \( Q_m \) be as in (2.2). Then \( n \pmod{p^m} \in B_{m-1} \) if and only if \( N \pmod{p^m} \in Q_m(B_{m-1} + 1) \).
(b) Let \( p^\alpha || k + 1 \). Since \( k + 1 \leq p^{m+1} \), we must have \( \alpha \leq m + 1 \). If \( \alpha = m + 1 \), then \( B_{m-1} = \{p^m - 1\} \), and hence \( n \equiv -1 \pmod{p^m} \). Now suppose \( \alpha \leq m \), and let \( Q_m \) be as in (2.3). Then

\[
N \equiv \frac{n + 1}{p^\alpha} Q_m \pmod{p^m}.
\]

Now if \( r \in B_{m-1} \), then \( r \pmod{p^\alpha} \in B_{\alpha-1} \); thus, \( r \equiv -1 \pmod{p^\alpha} \), so all the elements of \( B_{m-1} + 1 \) are divisible by \( p^\alpha \). Hence,

\[
N \pmod{p^m} \in Q_m \frac{B_{m-1} + 1}{p^\alpha}.
\]

In the particular case \( k + 1 = p^{m+1} \), the congruence \( n \equiv -1 \pmod{p^m} \) is always satisfied, and we do not need to include the modulus \( p^m \) in the sieving process. So in the case of \( k + 1 \) being composite, we can modify our earlier algorithm by changing step (3) of part (a) to:

3. If \( p \nmid k + 1 \), compute \( Q_m \) of (2.2) and load the residues \( Q_m(r_1 + 1), Q_m(r_2 + 1), \ldots, Q_m(r_q + 1) \pmod{p^m} \) into the ring of modulus \( p^m \).

If \( p^\alpha || k + 1 \) and \( \alpha \leq m \), compute \( Q_m \) of (2.3) and load the residues \( Q_m(r_1 + 1)/p^\alpha, Q_m(r_2 + 1)/p^\alpha, \ldots, Q_m(r_q + 1)/p^\alpha \pmod{p^m} \) into the ring of modulus \( p^m \).

If \( k + 1 = p^{m+1} \), then do not sieve on \( p^m \).

Part (b) of the algorithm is changed as follows. Since \( (g(k) + 1)/(k + 1) \geq (2k + 2)/(k + 1) = 2 \), we start the search at 2. Once a certain range has been sieved, test each solution candidate \( s \) by putting \( S = (k + 1)s - 1 \) and determining for each \( p \leq k \) that \( [S/p^m] \pmod{p} \geq a_m \). The least value of \( S \) for which this holds is \( g(k) \).

3. The table

In Table 1 we give all the values for \( g(k) \) for \( k \leq 140 \). To get some idea of how long this took to do, we point out that OASiS required 2 hours 48 minutes to compute \( g(111) \); it required about 11 days 11 hours to compute the largest value found, \( g(139) \); and it took 5 days 1 hour to compute \( g(112) \). For these last two values, the sieving times slightly exceeded the expected computation times of approximately 10 days and 4 days 21 hours, respectively (based on a rate of 215 000 000 trials per second). The reason for this is that OASiS verified the contents of its rings every hour, and each such checkpoint required around 9 minutes for \( g(139) \) and 2 minutes for \( g(112) \). The checkpoints for \( g(139) \) took significantly more time than those for \( g(112) \), since \( k = 139 \) required more congruences, so there were more rings to verify. We note here that in the cases of \( k = 111 \) and \( k = 139 \), sieving for \( h(k) \) did in fact achieve a speedup of roughly \( k + 1 \) relative to the expected time of sieving for \( g(k) \).

In [4], Erdös pointed out that the values of \( g(k) \) appear to grow much faster than the lower bound \( k^{1+c} \) given in [2], where \( c \) is a positive constant. Our computations seem to confirm this.
\begin{table}
\centering
\begin{tabular}{|c|c|c|c|c|c|}
\hline
\(k\) & \(g(k)\) & \(g(k)\) & \(g(k)\) & \(k\) & \(g(k)\) \\
\hline
2  & 6 & 49 & 38074099 & 96 & 5589371247 \\
3  & 7 & 50 & 4302206 & 97 & 104141995747 \\
4  & 7 & 51 & 13927679 & 98 & 10628330723 \\
5  & 23 & 52 & 366847 & 99 & 5675499 \\
6  & 62 & 53 & 79221239 & 100 & 3935600486 \\
7  & 143 & 54 & 7638454 & 101 & 2128236159983 \\
8  & 44 & 55 & 53583095 & 102 & 175209712494 \\
9  & 159 & 56 & 17868986 & 103 & 5092910127863 \\
10 & 46 & 57 & 34296443 & 104 & 6003175578749 \\
11 & 47 & 58 & 4703099 & 105 & 4753399456493 \\
12 & 174 & 59 & 108178559 & 106 & 4888983522367 \\
13 & 2239 & 60 & 93851196 & 107 & 6260627365379 \\
14 & 239 & 61 & 2237874623 & 108 & 9746385386989 \\
15 & 719 & 62 & 254322494 & 109 & 73243091349869 \\
16 & 241 & 63 & 157776319 & 110 & 94794806842238 \\
17 & 5849 & 64 & 266194499 & 111 & 222261611307119 \\
18 & 2098 & 65 & 174133871 & 112 & 90200708362489 \\
19 & 2099 & 66 & 25013442 & 113 & 517968108138869 \\
20 & 43196 & 67 & 673750867 & 114 & 517968108138869 \\
21 & 14871 & 68 & 6433664693 & 115 & 12714356616655615 \\
22 & 19574 & 69 & 237484869 & 116 & 4112143718554871 \\
23 & 35423 & 70 & 549177974 & 117 & 10584753118053749 \\
24 & 193049 & 71 & 3184709471 & 118 & 3781786358757119 \\
25 & 2105 & 72 & 4179979724 & 119 & 598228285941119 \\
26 & 36287 & 73 & 15780276223 & 120 & 260509131365372 \\
27 & 1119 & 74 & 19942847999 & 121 & 404087677322873 \\
28 & 284 & 75 & 48899668971 & 122 & 115598852533247 \\
29 & 240479 & 76 & 16360602718 & 123 & 714066520746239 \\
30 & 58782 & 77 & 2198202863 & 124 & 2820466143999 \\
31 & 341087 & 78 & 950337359 & 125 & 3988617067133 \\
32 & 371942 & 79 & 29154401359 & 126 & 5614007242751 \\
33 & 6459 & 80 & 43228410965 & 127 & 60503616486413 \\
34 & 69614 & 81 & 6599930719 & 128 & 14320632355808 \\
35 & 37619 & 82 & 1101163607 & 129 & 3134792099837 \\
36 & 152188 & 83 & 797012560343 & 130 & 7984603413422 \\
37 & 152189 & 84 & 95695473244 & 131 & 3249072073157063 \\
38 & 487343 & 85 & 449488751711 & 132 & 96965971239157 \\
39 & 767919 & 86 & 328151678711 & 133 & 1558724612351169 \\
40 & 85741 & 87 & 39419852119 & 134 & 621248003653094 \\
41 & 3017321 & 88 & 94923115999 & 135 & 3157756005623 \\
42 & 96622 & 89 & 3524996442239 & 136 & 4138989693368 \\
43 & 24041599 & 90 & 2487760912090 & 137 & 951598054985213 \\
44 & 45043199 & 91 & 739416801247 & 138 & 745504491090939 \\
45 & 9484095 & 92 & 2380889434844 & 139 & 2597202763644319 \\
46 & 692222 & 93 & 577593151999 & 140 & 908985422866845 \\
47 & 232906799 & 94 & 107706126974 &  &  \\
48 & 45375224 & 95 & 71573860223 &  &  \\
\hline
\end{tabular}
\end{table}
ACKNOWLEDGMENTS

The authors wish to thank Carole Lacampagne and John Selfridge for bringing this problem to their attention and suggesting the use of sieving techniques for the computation of $g(k)$.

BIBLIOGRAPHY


DEPARTMENT OF COMPUTER SCIENCE, UNIVERSITY OF MANITOBA, WINNIPEG, MANITOBA, R3T 2N2, CANADA

E-mail address, R. Scheidler: scheidl@silver.umanitoba.ca
E-mail address, H. C. Williams: hugh_williams@csmail.cs.umanitoba.ca