Quasi-perfect numbers have at least 8 prime divisors

Bernd Zemann

2023–08–04

Abstract

Quasi-perfect numbers satisfy the equation $\sigma(N) = 2 \cdot N + 1$, where σ is the divisor summatory function. By computation, it is shown that no quasi-perfect number has less than 8 prime divisors. For testing purposes, quasi-multiperfect numbers are examined also.
The author is not affiliated to any academic institution and does not claim that their work is original. [1].

[1] The author can be contacted by email (zb4ng@arcor.de) or their GitHub page [1]

1 Introduction

As of today, it is unknown, whether quasi-perfect numbers exist. In 1982, Hagis and Cohen [9] described an algorithm which was used to prove that no quasi-perfect number has less than 7 prime divisors.
By implementing their algorithm on a modern desktop PC, we are able to extend this result: We show that no number N divisible by 3 and $\omega(N) = 7$ is quasi-perfect. By Theorem 2 of [9] and earlier work of Kishore [11], we conclude:

Theorem 1.1. *Let $N \in \mathbb{N}$ with $\omega(N) \leqslant 7$. Then N is not quasi-perfect.*

In order to test the software, a more general equation was investigated:

$$\sigma(N) = k \cdot N + 1 \tag{1}$$

where k is an integer usually greater than 2. The numbers N that satisfy this equation are called quasi-multiperfect, but we also use the term quasi-k-perfect. Only for $3 \leqslant k \leqslant 5$, the computations are accessible by modest means, since for $k > 5$, a simple consideration shows that $\omega(N) \geqslant 9$ and the algorithm would take too much time in any case.
In this way, no solutions to 1 were found, but bounds for $\omega(N)$ can be given. There is a special problem with $k = 3$, as will be described later.
On the whole, we have following result [2] :

Theorem 1.2. *For $2 \leqslant k \leqslant 5$, then $\omega(N)$ is greater or equal to numbers given in the following table 1*

k	even	odd
2	n/a	8
3	2	10
4	10	21*
5	9	54*
6	10	141*
7	14*	372*

Table 1: Table of lower bounds for $\omega(N)$ depending on k. The values marked with * come from a simple estimation.

1.1 Notation and Preliminaries

In the following, N always means a natural number, having the factorization

$$N = \prod_{j=1}^{r} p_j^{a_j} \tag{2}$$

[2] For additional results for quasi-multiperfect numbers refer to [16] and [12]

where $r \in \mathbb{N}$ and $p_1 < \ldots < p_r$ are primes. Additionally p is always a prime
We may also use the following notation : Let $S := \{\cdot, \beta\}$ be a symbol set. Closely linked to the prime factors p_j and exponents a_j of N, a vector $\boldsymbol{\lambda} = \{\lambda_1, \ldots, \lambda_r\} \in S^r$ is defined.

Some common number-theoretic functions are used throughout the text with their usual notation:

- $\sigma(N) := \sigma_1(N) := \sum_{d \mid N} d$ is the sum of divisors.
- $\omega(N)$ is the number of primes dividing N.
- $\left(\frac{x}{p}\right)$ with $x \in \mathbb{Z}$ is the Legendre symbol.

In particular, we have,

$$\sigma(p^j) = \sum_{j=0}^{a} p^j = \frac{p^{a+1} - 1}{p - 1} \tag{3}$$

In addition, we define:

$$h(N) := \frac{\sigma(N)}{N}$$

and some variations of this function:
For a prime p, we set

$$h_\infty(p) := \frac{p}{p-1}$$

If a symbol vector $\boldsymbol{\lambda}$ is assigned to N, we define:

$$h_{\max}\left(p_j^{a_j}\right) := \begin{cases} h_\infty\left(p_j\right) & \text{if } \lambda_j = \beta \\ h\left(p_j^{a_j}\right) & \text{otherwise} \end{cases}$$

Sometimes, we also use $h_{\min}$ instead of h.

1.2 Technical Details of the Program

The program was written in C++ (C++2017 standard) and makes extensive use of the multi-precision libraries GMP [8] for integer arithmetic and MPFR [13] for multi-precision floating point arithmetic.
To a minor degree NTL [15] by Victor Shoup and a deterministic primality testing algorithm [5] is employed.
As a wrapper for MPFR and GMP as well as for various other purposes, the Boost library [2] is linked.

In the next section, some useful properties of quasi-multiperfect numbers are presented, subsequently, the algorithm and results are described insofar as necessary.
The source code of the associated computer program can be found on GitHub [1].

2 Quasi-k-perfect numbers

In this section, we want to examine some properties of quasi-k-perfect numbers and how the algorithm in [9] can be applied in this case.

In this section, we assume that N is quasi-k-perfect, i.e. satisfies 1.

As an aside, note that quasi-1-perfect numbers are exactly the primes.

2.1 Feasible Exponents

We begin with a generalization to some properties from [9], [10] and [4]:

Lemma 2.1. *If one of the following conditions is satisfied*

1. *k is even.*
2. *k is odd and N is even.*

, then

$$N = 2^a M^2$$

, where M is odd and a can be zero for the first condition. In particular, for $k = 2$, we have the familiar result that N is an odd square.

Proof. 1. Let $p^b \parallel N$ with p and b odd. Then

$$\sigma(p^b) = \sum_{j=0}^{b} p^j \equiv (b+1) \equiv 0 \mod 2$$

But $\sigma(N)$ is odd.

2. similar

□

We now seek to generalize the notion of feasible exponents:

Lemma 2.2. *Let $N := 2^{a_1} N'$ be a quasi-k-perfect number, N' odd and $p^a \parallel N'$. Write $k := 2^b \cdot k'$, where $b \geqslant 0$ is any integer and k' is odd. Let q be a prime divisor of k'. Then:*

1. *We have,*

$$(k, \sigma(p^a)) = 1$$

2. *If $a_1 + b > 0$, and r is a prime dividing $\sigma(N)$ coprime to k' then*

$$\left(\frac{-2^{a_1} k}{r}\right) = \left(\frac{-2^{a_1+b} k'}{r}\right) = 1 \tag{4}$$

3. *If* $p \equiv 1 \mod q$*, then* $a \not\equiv -1 \mod q$*.*

4. *If* $p \equiv -1 \mod q$*. then* a *is even.*

Proof. 1. Obvious.

2. By 2.1,
$$k \cdot N = 2^{a_1+b} k' M^2 \equiv -1 \mod r$$
for some integer M. Multiplying by $2^{a_1+b}k'$ proves the hypothesis.

3. If $a \equiv -1 \mod q$,
$$\sigma(p^a) \equiv a + 1 \equiv 0 \mod q$$
, which is impossible.

4. Assume a is odd:
$$\sigma(p^a) \equiv \sum_{j=0}^{a} (-1)^j \equiv 0 \mod q$$
As before this gives a contradiction.

□

2.2 Constraints

For the algorithm, a lower bound N_0 for quasi-k-perfect numbers is needed. For $k = 2$, we use $N_0 = 10^{20}$.

By a simple SAGE program, we confirmed that for $k \geqslant 2$ there are no numbers
$$N \leqslant 10^8$$
, s.t.
$$\sigma(N) = \pm 1$$
, hence no quasi-k-perfect numbers and in this case $N_0 = 10^8$.

Furthermore, by taking into account that
$$h(N) = \prod_{j=1}^{r} h\left(p_j^{a_j}\right) \leqslant \prod_{j=1}^{r} h\left(Q_j^{\infty}\right) = \prod_{j=1}^{r} \frac{Q_j}{Q_j - 1} =: A_r \tag{5}$$
, where Q_j is the sequence of primes $2, 3, , \ldots$, we see that we can ignore the N with
$$\omega(N) = r$$
if $A_r < k$.

The following table shows the smallest of value of ω , a hypothetical quasi-k-perfect can have according to 5:

k	even	odd
2	1	3
3	2	8
4	4	21
5	6	54
6	9	141
7	14	372

Table 2: Table of lower bounds for $\omega(N)$ depending on k implied by 5

2.3 The prime bounds

Remember that we only search for quasi-k-perfect numbers N with $\omega(N) = r$ for some fixed r. The basic idea for our algorithm is that you have some number M with $\omega(M) < r$ and want to find a bound for a prime p s.t. Mp^a can be a divisor of N.
In order to achieve this, we can just reuse - *mutatis mutandis* - the lemmas below from [9] and earlier work ([10] , [14]):

Lemma 2.3. *(Jerrard and Temperley, [10]) Let $q := p_{r-1}$ and $p := p_r$,hence $q < p$. and $N = Mp^{a_{r-1}}q^{a_r}$. Moreover, write $F(N) := k \cdot N - \sigma(N)$. Then*

$$\frac{kN}{F(N)} - \frac{1}{q} < p < \frac{kN}{F(N)}$$

From this Lemma 2 in [9] is derived, of which we use a modified version to compute bounds for the biggest prime factor p_r:

Lemma 2.4. *(Hagis and Cohen , [9]) Let F, E and U are defined as in [9],*

$$R := \frac{k}{k-F}$$

$$L := R - \frac{kFU\,(k-F+FU)}{k-F}$$

, then

$$L - (L - (q+k)^{-1})^{-1} \leqslant p < R \tag{6}$$

The bounds for smaller prime factors p_j with $2 \leqslant j < r$ come from Lemma 1 in [9]:

Lemma 2.5. *(Hagis and Cohen , [9], somewhat modified) Let s be an index with $1 \leqslant s \leqslant r-2$, $M := \prod_{j=1}^{s} p_j^{a_j}$ and $\boldsymbol{\lambda} \in S^s$ a symbol vector. Let $B := \frac{k}{h_{min}(M)}$ and $D := \frac{h_{max}(M)}{k}$. Then*

$$p_{s+1} > \frac{\sqrt{4B-3}+1}{2\,(B-1)} \tag{7}$$

and

$$p_{s+1} > \frac{1}{1 - D^{\frac{1}{t}}} \tag{8}$$

, where $t := r - s$.

2.4 Special k

For $k = 4$, we have

Theorem 2.6. *If* N *is quasi-*4*-perfect, then*

$$N = 2^a M^2$$

, where $a \in \{0, 1\}$ *and* M *is odd*

Proof. By 2.1, it suffices to disprove $a > 1$. We show that in this case,

$$\left(\frac{-2^a k}{r}\right) = \left(\frac{-2^a}{r}\right) = -1$$

for some divisor r of $\sigma(N)$.
If a is odd, since $\sigma(2^a) \equiv -1 \mod 8$, there must be some $r \mid \sigma(N)$ with $r \equiv 5, 7 \mod 8$, hence

$$\left(\frac{-2^a}{r}\right) = \left(\frac{-1}{r}\right) = -1$$

If a is even, since there must be r with $r \equiv 3 \mod 4$, we have

$$\left(\frac{-2^a}{r}\right) = \left(\frac{-1}{r}\right) = -1$$

□

3 Description of the Algorithm

Based on the assertions of the previous section, a computer program was implemented and executed. As mentioned earlier, the algorithm is described in [9]:

3.1 Table of Feasible Exponents

By the explanation in [9] for $k = 2$ and 2.1 for $k > 2$, only certain exponents (which are called feasible) of some prime can be quasi-k-perfect.

In order to create a table of feasible exponents for the parameter k in question, the prime factorization of $\sigma(p^a)$ is needed. Taking 3 into account, the following factorization tables for $p^a - 1$ were used:

- The Cunningham project [6], if $p \leqslant 11$, see also [3]
- An updated version of the factor table of Richard P. Brent, maintained by a different author [7] , for $11 < p < 10000$. [3]

Since [7] contains only prime factors up-to 10^9, smaller factors had to be found by trial division. Afterwards for every prime $p < 10000$ a list of feasible exponents a was created with the condition $p^a < 10^{20}$.

3.2 Iteration

Now, we give a description of the main part of the program: for this purpose, it suffices to confine ourselves to the situation of Thm. 1.1 ($k = 2$ and $r = 7$):

- Fix $p_1 = 3$.
- Iteration according to the following scheme:
  - If $j < r - 1$ and the iteration is at the prime factors $p_1, \ldots, p_j$ with exponents $a_1, \ldots, a_j$ and some vector λ, then p_{j+1} is the smallest prime satisfying 7 and a_{j+1} is the smallest feasible exponent.
  - Set $j \to j + 1$
  -
- The prime p_j is generated by iterating over an interval with bounds dependent on the prime components $p_k^{a_k}$ with $1 \leqslant k < j$ by using 7 and 8 for $j < r$ and 6 for $j = r$. The exponent a_j is iterated over all feasible exponents and then β_j. Concerning the aforementioned vector λ, we define $\lambda_j = \beta$ iff $a_j = \beta_j$.

[3] The original website was unavailable at the time, when this text was written.

- If we find p_r in the previous step, we have a set of candidates for a quasi-perfect number:

$$N = \prod_{l=1}^{r} p_l^{c_l}$$

  where $c_l = a_l$ if $\lambda_l = \cdot$ and c_l ranges over all integers $\geqslant \beta_l$ if $\lambda_l = \beta$. In addition, c_r ranges over all integers.
  These candidates are then checked, if any of them is quasi-perfect.

- In two cases for $k = 2$, the previous step was inconclusive and it was confirmed with SAGE that none of the concerning candidates were quasi-perfect (see A).

4 Results

A quick search with SAGE showed that there is no solution of

$$\sigma(N) = k \cdot N \pm 1$$

for $N \leqslant 10^8$ and any $k \geqslant 2$.

4.1 $k = 2$

For $k = 2$, quasi-perfect numbers N with $\omega(N) = 7$ were searched for, and it was established that none exist.
Moreover, we have the following running times (for $4 \leqslant \omega(N) \leqslant 6$ measured on 2022-02-20):

	length/$\omega(N)$	time
	4	0.00950 secs
	5	0.168 secs
	6	20.5 secs
	7	$1.15 \cdot 10^6$ secs

The time for $\omega(N) = 6$ translates to around 13 days. A (very rough) extrapolation for $\omega(N) = 8$ gives a running time of at least 2000 years.

4.2 $k > 2$ [4]

It turned that - taking into account the limited computing power available to the author - we are restricted to $\omega(N) \leqslant 7$. Hence by 2.2, the following calculations were done.

4.2.1 $k = 3$

This case has a peculiarity, since

$$h_\infty(2) \cdot h_\infty(3) = 3$$

and our bounding method doesn't work properly if $6 \mid N$, because we cannot exclude any prime p_3 - however big - dividing N, even if we choose -say - $\omega(N) = 3$.
For odd N, we could check that $\omega(N) \geqslant 10$ and improve the bound from table 2.2.

4.2.2 $k = 4$

Search for even quasi-4-perfect numbers N: None were found with

$$\omega(N) \leqslant 9$$

[4]Some of these results were already described by the authors Meng Li and Min Tang in [16] and [12].

A Special cases

This section contains the SAGE notebook that is used to deal with the cases that could not be handled by the software.

quasiperfect_special

December 9, 2022

1 Exponents for special vectors

Here we disprove the existence of quasi-perfect numbers in the two remaining cases: the prime factorizations are given as lists.

File: 132 [08:51:24] - QuasiPerfect::calculate(): bounds are satisfied: 4300 at iteration 29477

#[08:51:24] - prvec: [(3,44b),(5,22),(17,18b),(257,10b),(66161,4b),(10356029,2b),(21015221,2b)]

File: 146 [09:00:39] - QuasiPerfect::calculate(): bounds are satisfied: 4575 at iteration 31131

[09:00:39] - prvec: [(3,44b),(5,30b),(17,18b),(263,10b),(9601,6b),(7505611,4b),(13084021,4b)]

[ ]:
```
We need the following functions:
```

[1]:
```
def h(n):
    return sigma(n)/n
def hinf(p):
    return p/(p-1)
def h_lst(flst):
    result = 1
    for p,exp in flst:
        if exp == 'inf':
            result *= hinf(p)
        else:
            result *= h(p^exp)
    return result
def getlstValue(flst):
    result = 1
    for p,exp in flst:
        result *= p^exp
    return result
```

1st case: prime vector: [(3,44b),(5,22),(17,18b),(257,10b),(66161,4b),(10356029,2b),(21015221,2b)]
We check that for p1 = 3 the exponent a1 = 44 cannot occur, since h(N) is always smaller than 2.

[2]:
```
flst1 = [(3,44),(5,22),(17,18),(257,10),(66161,4),(10356029,2),(21015221,2)]
flst2 =␣
 ↪[(3,44),(5,22),(17,'inf'),(257,'inf'),(66161,'inf'),(10356029,'inf'),(21015221,'inf')]
print(flst1)
```

```
print(flst2)
print ("h(flst1) < 2 ?", h_lst(flst1) < 2)
print ("h(flst2) < 2 ?",  h_lst(flst2) < 2)
```

```
[(3, 44), (5, 22), (17, 18), (257, 10), (66161, 4), (10356029, 2), (21015221,
2)]
[(3, 44), (5, 22), (17, 'inf'), (257, 'inf'), (66161, 'inf'), (10356029, 'inf'),
(21015221, 'inf')]
h(flst1) < 2 ? True
h(flst2) < 2 ? True
```

hence we take a1 >= 52 (52 being the next feasible exponent for 3)
Similarly for a6 = 2:

[3]:
```
flst3 = [(3,52),(5,22),(17,18),(257,10),(66161,4),(10356029,2),(21015221,2)]
flst4 =␣
  ↪[(3,'inf'),(5,22),(17,'inf'),(257,'inf'),(66161,'inf'),(10356029,2),(21015221,'inf')]
print(flst3)
print(flst4)
print ("h(flst3) < 2 ?", h_lst(flst3) < 2)
print ("h(flst4) < 2 ?",  h_lst(flst4) < 2)
```

```
[(3, 52), (5, 22), (17, 18), (257, 10), (66161, 4), (10356029, 2), (21015221,
2)]
[(3, 'inf'), (5, 22), (17, 'inf'), (257, 'inf'), (66161, 'inf'), (10356029, 2),
(21015221, 'inf')]
h(flst3) < 2 ? True
h(flst4) < 2 ? True
```

hence we take a6 >= 4. Finally, we test a7 = 2:

[51]:
```
flst5 = [(3,52),(5,22),(17,18),(257,10),(66161,4),(10356029,4),(21015221,2)]
flst6 =␣
  ↪[(3,'inf'),(5,22),(17,'inf'),(257,'inf'),(66161,'inf'),(10356029,'inf'),(21015221,2)]
print(flst5)
print(flst6)
print ("h(flst5) < 2 ?", h_lst(flst5) < 2)
print ("h(flst6) < 2 ?",  h_lst(flst6) < 2)
```

```
[(3, 52), (5, 22), (17, 18), (257, 10), (66161, 4), (10356029, 4), (21015221,
2)]
[(3, 'inf'), (5, 22), (17, 'inf'), (257, 'inf'), (66161, 'inf'), (10356029,
'inf'), (21015221, 2)]
h(flst5) < 2 ? True
h(flst6) < 2 ? True
```

hence we take a7 >= 4

[60]:
```
flst7 = [(3,52),(5,22),(17,18),(257,10),(66161,4),(10356029,4),(21015221,4)]
```

```
flst8 =␣
  ↪[(3,'inf'),(5,22),(17,'inf'),(257,'inf'),(66161,'inf'),(10356029,'inf'),(21015221,4)]
print(flst7)
print(flst8)
print ("h(flst7) > 2 ?", h_lst(flst7) > 2)
print ("h(flst8) > 2 ?",  h_lst(flst8) > 2)
N= getlstValue(flst7)
print (sigma(N) - 2*N)
```

```
[(3, 52), (5, 22), (17, 18), (257, 10), (66161, 4), (10356029, 4), (21015221,
4)]
[(3, 'inf'), (5, 22), (17, 'inf'), (257, 'inf'), (66161, 'inf'), (10356029,
'inf'), (21015221, 4)]
h(flst7) > 2 ? True
h(flst8) > 2 ? True
860383253136690301496773880895548436723440020589898823595972125069936109953784178
822581868101022794571800154417435609800343185726221527159472 5
```

In the last step, we have shown that h(N)>2 if a7 >= 4 and that the smallest of these values isn't qp. Therefore there are no qp numbers with the given prime factors!

2nd case: prime vector: [(3,44b),(5,30b),(17,18b),(263,10b),(9601,6b),(7505611,4b),(13084021,4b)]
As in the 1st case, a1 = 44 is not possible:

[65]:
```
flst1 = [(3,44),(5,30),(17,18),(263,10),(9601,6),(7505611,4),(13084021,4)]
flst2 =␣
  ↪[(3,44),(5,'inf'),(17,'inf'),(263,'inf'),(9601,'inf'),(7505611,'inf'),(13084021,'inf')]
print(flst1)
print(flst2)
print ("h(flst1) < 2 ?", h_lst(flst1) < 2)
print ("h(flst2) < 2 ?",  h_lst(flst2) < 2)
```

```
[(3, 44), (5, 30), (17, 18), (263, 10), (9601, 6), (7505611, 4), (13084021, 4)]
[(3, 44), (5, 'inf'), (17, 'inf'), (263, 'inf'), (9601, 'inf'), (7505611,
'inf'), (13084021, 'inf')]
h(flst1) < 2 ? True
h(flst2) < 2 ? True
```

hence a1 >= 52. But now we can show that the smallest possible value for h(N) is greater than 2.

[6]:
```
flst3 = [(3,52),(5,30),(17,18),(263,10),(9601,6),(7505611,4),(13084021,4)]
print(flst3)
print ("h(flst3) > 2 ?", h_lst(flst3) > 2)
```

```
[(3, 52), (5, 30), (17, 18), (263, 10), (9601, 6), (7505611, 4), (13084021, 4)]
h(flst3) > 2 ? True
```

Also the corresponding number flst3 is not qp:

[68]:
```
N= getlstValue(flst3)
print (sigma(N) - 2*N)
```

```
35594511883585858678825834554203011975967913283688988938878828986083391125190951
5942894120569993932548085856712302837030851976962781892542221129038425
```

and so we have shown that there are no qp numbers in this case!

References

[1] Author - Github Page. https://github.com/sikefield3/quasiperfect. Accessed: 2023-04-10.

[2] Boost. https://www.boost.org. Accessed: 2021-12-28.

[3] John Brillhart et al. *Factorizations of BN ± 1: B = 2, 3, 5, 6, 7, 10, 11, 12 up to high powers*. American Mathematical Society, 1988.

[4] Paolo Cattaneo. Sui numeri quasiperfetti. *Boll. Unione Mat. Ital., III. Ser.*, 6:59–62, 1951.

[5] David Cleaver. MPZ APRCL. https://sourceforge.net/projects/mpzaprcl/. Accessed: 2021-12-28.

[6] Cunningham Project. The third edition of the Cunningham book. https://homes.cerias.purdue.edu/~ssw/cun/third/index.html. Accessed: 2022-09-09.

[7] Factor collection. https://web.archive.org/web/20210212010716/myfactorcollection.mooo.com:8090/original.html. Accessed: 2021-09-09.

[8] GMP. Gnu multiple precision arithmetic library (gmp). https://gmplib.org/. Accessed: 2021-12-28.

[9] Peter Hagis and Graeme L. Cohen. Some results concerning quasiperfect numbers. *Journal of the Australian Mathematical Society. Series A. Pure Mathematics and Statistics*, 33(2):275–286, 1982.

[10] R. P. Jerrard and Nicholas Temperley. Almost perfect numbers. *Mathematics Magazine*, 46(2):84–87, 1973.

[11] Masao Kishore. Odd integers n with five distinct prime factors for which $2-10^{-12} < \sigma(n)/n < 2+10^{-12}$. *Mathematics of Computation*, 32(141):303–s12, 1978.

[12] Meng Li and Min Tang. On the congruence σ (n) $\equiv 1 \mod$ n, II. *Journal of Mathematical Research with Applications*, 34(2):155–160, 2014.

[13] MPFR. Gnu multiple precision floating-point reliable library (GNU MPFR). https://www.mpfr.org/. Accessed: 2021-12-28.

[14] Carl Pomerance. Odd perfect numbers are divisible by at least seven distinct primes. *Acta Arithmetica*, 25(3):265–300, 1974.

[15] Victor Shoup. NTL: A library for doing number theory. https://libntl.org/. Accessed: 2021-12-28.

[16] Min Tang and Meng Li. On the congruence σ (n) $\equiv 1 \mod$ n. *Journal of Mathematical Research with Applications*, 32(6):673–676, 2012.