22111

完全数

2604 石原慶次　2519 小西光柊　2521 鈴木望　2623 永屋和輝

要旨

私たちは，未解決問題である完全数に興味を持ち、それをテーマとして研究した。約数関数の規則性や倍積完全数の見つけ方について表計算ソフトなどを用いて調べた。1 から 100 万までの自然数に奇数の完全数は見つからなかった。そこで，奇数の完全数があると仮定して，これを式に表せることを調べた。その結果，この式を応用することで，倍積完全数を見つけることができた。今後も計算ソフトなどを活用して計算を進め，その式についての新しい条件を探していく。

１．目的

完全数に関する様々な謎を解明する。(具体的なものは、「４．研究方法」参照)

２．定義

・完全数

約数の総和が元の数の 2 倍になる数。

（ex）6：1+2+3+6=12=6×2

⇒6 は完全数

2021 年 12 月現在では、51 個の完全数が見つかっている。この完全数に関して様々な未解決問題が存在している。

・$\sigma(x)$

xの約数の総和を$\sigma(x)$と表す。また、$y=\sigma(x)$である関数を約数関数と呼ぶ。これには部分的な乗法性があり、互いに素な 2 整数a,bに関して、$\sigma(ab)=\sigma(a)\sigma(b)$が成り立つ。

３．使用ソフト

・Excel

・SageMath

４．研究方法

本研究では、主に次の 4 つのことをテーマに研究を行った。

（Ⅰ）奇数の完全数について

（Ⅱ）倍積完全数について

（Ⅲ）様々な数での約数関数について

（Ⅳ）約数関数の性質について

５．研究内容・結果

(Ⅰ)奇数の完全数について

(ⅰ)目的

現在、完全数は 51 個確認されているが、奇数のものは 1 つも発見されていない。よって本研究では、奇数の完全数を発見するために、その性質を探った。

(ⅱ)仮説

仮説として「奇数の完全数には、その約数に規則性や条件がある」と考えた。偶数の完全数には、2^p-1 が素数（メルセンヌ素数という）となるような素数pを用いて、$2^{p-1}(2^p-1)$ の形で表せるという性質がある。そのため、奇数の完全数にもそのように約数に規則性が存在するのではないかと考えたためである。

(ⅲ)結果・考察

先行研究より、奇数の完全数Lが存在すると仮定すると、Lは奇素数$p_1 \sim p_n$, qと、その指数$e_1 \sim e_n, a$ を用いて、$L=q^a \times {p_1}^{2e_1} \times {p_2}^{2e_2} \times \ldots \times {p_n}^{2e_n}$ と表せる（nは自然数、以下同様）ことが分かっている。そのため、$\frac{\sigma(L)}{L}$は

$$\frac{\sigma(L)}{L}=\frac{\sigma(q^a)\times\sigma({p_1}^{2e_1})\times\sigma({p_2}^{2e_2})\ldots\times\sigma({p_n}^{2e_n})}{q^a\times{p_1}^{2e_1}\times{p_2}^{2e_2}\times\ldots\times{p_n}^{2e_n}}$$

で表せる（以下式①とする）。この式①の値が 2 になれば完全数であるので、先行研究から次の 2 つの条件が成り立つと知り、以下のように示した。

〈条件〉

（A） $q\equiv a\equiv 1(mod4)$

（B） 素数$p_1\sim p_n$ は偶数乗される。

（A）、（B）はともに、式①の分子に 1 つだけ 2 が現れるための条件である。式①の分母は奇数、分子は偶数、値は 2 なので、$\sigma(L)=2L\equiv 2(mod4)$ である。$p_1\sim p_n$の約数の総和が奇数である必要があるため、$p_1\sim p_n$を偶数乗する（B）。また、分子は偶数になるので、分母の奇素数のうち一つだけは、約数の総和が 4 で割ると、2 余る数になる。よって、その奇素数の指数は、奇数となる。その奇素数が式①では、qであり、その指数はaである。

$q\equiv 3(mod4)$ のとき、$\sigma(q^a)$ は常に 4 の倍数となるので、適さない。

一方、$q\equiv 1(mod4)$ のとき、$\sigma(q^a)$は、$a\equiv 1(mod4)$ の時に 4 で割ると、2 余る数になる。

よって、$q\equiv a\equiv 1(mod4)$ （A）

以上より、奇数の完全数が持つ約数の条件を確認したが、これらを満たす$q^a,{p_1}^{2e_1},{p_2}^{2e_2},\ldots,{p_n}^{2e_n}$を見つけ出すのは極めて困難であると判断したため、奇数の完全数を発見するには、別の方法が必要だと考えた。

ここで、我々は、奇数の完全数を見つけることから、作り出す方向に発想を転換し、研究を進めた。その結果、「始めに自然数を設定し、分母が全て約分によって消えるように$q^a,{p_1}^{2e_1},{p_2}^{2e_2},\ldots,{p_n}^{2e_n}$を決定する」という方法を考案した。

〈例〉

$$\frac{2\times 7}{13}\times\frac{3\times\boxed{19}}{7^2}\times\frac{3\times 127}{\boxed{19^2}}\times\cdots\times\frac{\sigma(k^{2e})}{k^{2e}}$$

① ② ③ ④

① 始めに素数（以後、始めの値と呼ぶ）を条件（A）に従い設定する。上図では、13^1（図中五角形）としている。

② 分子には、①で立てた自然数の約数の総和を立てる。上図では、13 の約数の総和である 14 つまり 2×7（図中丸）となっている。
また、次の分数の分母には、7 と約分して消えるように${p_2}^{2e_2}$ を立てる。

③ ②を分母が約分によって消えるまで同じように繰り返す。

④ 操作を終えた時、分母に立てた値の積が求める奇数の完全数Lである。

このような操作（以後操作 A と呼ぶ）をすることで、奇数の完全数を発見できると考えた。しかし、さまざまな初めの値を設定し、コンピューターで計算しても、操作 A は終了することが無かった。原因として、$p_1\sim p_n$は、分母で偶数乗の形で存在しなければならないが、上の例では、一つしか約分されておらず、残りは別の奇素数の約数の総和にある素因数で約分する必要がある。このことから、この操作 A を用いても奇数の完全数を発見することは難しいと判断した。

（Ⅱ）倍積完全数について

（ｉ）目的

倍積完全数とは、約数の総和が元の数の自然数倍になる数のことを指す。

また、約数の総和が元の数のk倍になる時、その元の数をk倍完全数と呼ぶ。

(例) 120：1+2+…+60+120=360

=120×3⇒120 は 3 倍完全数

kの最大値、倍積完全数の無限性や規則性等は明らかになっていない。

我々はまず、Excel を用いて倍積完全数にどのような数があるのかを調べた。結果は次のようだった。

n	$\frac{\sigma(n)}{n}$
6	2
28	2
120	3
496	2
672	3
8128	2
30240	4
32760	4
523776	3

1≦n≦1000000 で、倍積完全数は 9 個存在した。しかし、1000000 個の数を調べて、9 個しか見つからないのでは明らかに効率が悪いと感じた。そこで、より少ない試行回数で、より多くの倍積完全数を見つける方法について研究を行った。

(ⅱ) 仮説

この研究での仮説は、奇数の完全数と同じように、「倍積完全数にはその約数に規則性や条件がある」とした。倍積完全数の効率的な発見方法を調べるために、約数に着目し、規則性を見出すことで、調べる数の量を減らせるのではないかと考えた。

(ⅲ) 結果・考察

全ての自然数Nは

$$N = {p_1}^{e_1} \times {p_2}^{e_2} \times \ldots \times {p_n}^{e_n}$$

と表せる。よって$\frac{\sigma(N)}{N}$は次のように表される。

$$\frac{\sigma(N)}{N} = \frac{\sigma({p_1}^{e_1}) \times \sigma({p_2}^{e_2}) \times \ldots \times \sigma({p_n}^{e_n})}{{p_1}^{e_1} \times {p_2}^{e_2} \times \ldots \times {p_n}^{e_n}}$$

Nが倍積完全数である場合、この値は自然数となる。

ここで我々は、操作 A を応用することで、倍積完全数を発見できるのではないかと考えた。式①の値が 2 になるように分母を消す操作 A と、倍積完全数の式の値を自然数となるようにするという点で類似していると考えたためである。操作 A との違いとしては、条件(A)、(B)が成り立たなくてよいため、それをもとに操作を行った。その一例が次の図である。

$$\frac{\sigma(N)}{N} = \frac{3 \times \boxed{5} \times 17}{2^7} \times \frac{2 \times 3^2}{17} \times \frac{2 \times 3}{\boxed{5}} \times \frac{11^2}{3^4} \times \frac{7 \times 19}{11^2} \times \frac{2^2 \times 5}{19} \times \frac{2^3}{7}$$

① ② ③ ④

$$= 5$$

① 始めに自然数を分母に立て、その分子に約数の総和を取る。ここでは例として${p_1}^{e_1} = 2^7$ と設定し、約数の総和は 255 つまり 3×5×17 である。(図中五角形)

② 次に分子にある 17 と約分して分母が消える分数を立てるので、分母は17^1であり、約数の総和は、18 つまり 2×3²である。(図中丸)

③ これを、全ての分母が約分によって消えるまで操作を行う。

このように操作を行うと分母は全て約分で消え、式の値は 5 となった。ここで分母の数の積を求めることで 5 倍完全数 14182439040 を作り出すことができた。

しかし、この方法では${p_1}^{e_1}$に代入する数によっては、倍積完全数ができない場合もあることも分かった。なぜならば、一度分母として立てた素因数をもう一度分母に立てざるを得なくなった

場合に、それ以上計算を進めることができなくなるためである。

これは、元の式

$$\frac{\sigma(N)}{N}=\frac{\sigma(p_1{}^{e_1})\times\sigma(p_2{}^{e_2})\ldots\times\sigma(p_n{}^{e_n})}{p_1{}^{e_1}\times p_2{}^{e_2}\times\ldots\times p_n{}^{e_n}}$$

が異なる$p_1{}^{e_1},p_2{}^{e_2},\ldots,p_n{}^{e_n}$に分けて示していることと、$\sigma(x)$の乗法性に反するからである。

我々は、SageMath という数式処理ソフトを用いて、$p_1{}^{e_1}$ を設定することで、そこから操作によって発見される倍積完全数を返すプログラムを作成した。右の画像がそれである。17、18 行目の m, e に値を代入することで、前述の①～③の操作を行う。

このプログラムを用いて、始めの数に様々な値を設定して倍積完全数を調べてみた。始めに設定する値は、その素因数が小さい方が分母が約分で消えやすいと予想したため、$p_1{}^{e_1}$は、2 や 3 といった小さな素因数の累乗を設定して調査を行った。

始めの値	出てきた完全数	倍	始めの値	出てきた完全数	倍
2^1	6	2	2^13	4.03031E+11	4
2^2	28	2	2^14	51001180160	3
2^3	32760	4	2^16	8589869056	2
2^4	496	2	2^18	1.37439E+11	2
2^6	8128	2	2^20	1.80258E+21	5
2^7	14182439040	5	2^30	2.30584E+18	2
2^8	459818240	3	2^60	2.65846E+36	2
2^9	142990848	4	2^88	1.91562E+53	2
2^10	1.36619E+13	5	3^6	1.49421E+16	4
2^11	5.18667E+11	5	3^10	6.08873E+15	4
2^12	33550336	2	7	32760	4

150 個の数を検証した結果、上の表にあるような 22 個の倍積完全数を発見することができた。Excel での調査では、100 万までの自然数を全て調べて 9 個の倍積完全数を発見したため、Excel での調査に比べると、遥かに効率が良いことが分かる。予想したように始めの値を小さな素数の累乗の数にすると倍積完全数をいくつか発見できた。また、見つかった倍積完全数の中で一番大きな数は、始めの値を2^{88}と設定して発見された 54 桁の数であるが、始めに設定する値の大きさと、発見される倍積完全数の大きさに関連は見いだせなかった。

```
sage: from __future__ import division
....: from typing import List
....: from multiset import Multiset
....: from sympy import symbols
....: from sympy import Function
....: from sympy import factorint
....: x, y, z, t = symbols('x y z t')
....: k, m, n = symbols('k m n', integer=True)
....: f, g, h = symbols('f g h', cls=Function)
....: def func(a:List[int],b:[int]):
....:     a_=a.copy()
....:     for i in a:
....:         if i in b:
....:             a_.remove(i)
....:             b.remove(i)
....:     return a_,b
....: m=2
....: e=100
....: A=[]
....: a=[]
....: for i in range(e):
....:     A.append(m)
....:     a.append(m)
....: S=(m^(e+1)-1)/(m-1)
....: B=factorint(S,multiple=True)
....: b=factorint(S,multiple=True)
....: m=max(B)
....: e=B.count(m)
....: x=Multiset(A)
....: y=Multiset(B)
....: while not(y>x):
....:     s=(m^(e+1)-1)/(m-1)
....:     B.extend(factorint(s,multiple=True))
....:     b.extend(factorint(s,multiple=True))
....:     for i in range(e):
....:         A.append(m)
....:         a.append(m)
....:     A,B=func(A,B)
....:     if set(A) == set():
....:         print("分母1:",a)
....:         a,b=func(a,b)
....:         print(b,"倍です")
....:         break
....:     C=[]
....:     c=[]
....:     C.extend(A)
....:     c.extend(a)
....:     C,c=func(C,c)
....:     c,B=func(c,B)
....:     if set(B) == set():
....:         print("分子がありません")
....:         break
....:     m=max(B)
....:     e=B.count(m)
....:     x=Multiset(A)
....:     y=Multiset(B)
....: else:
....:     print("分母2:",a)
....:     a,b=func(a,b)
....:     print(b,"倍です")
```

我々は今回倍積完全数を見つけることができなかった始めの値からも倍積

完全数を発見できないかと考え、新たなプログラムを作成した。倍積完全数が発見できない場合は、具体的には次のような場合である。

$$\frac{\sigma(n)}{n}=\frac{3^2\times 7}{2^5}\times\frac{2^3}{7}\times\frac{13}{3^2}\times\frac{2\times 7}{13}$$

上の例では、始めの値を${p_1}^{e_1}=2^5$とした時であるが、7 は一度分母に出てきた数であるにも関わらず、もう一度 7 を立てざるを得ない状況になっている。ここで分子に 2 が 5 個以上あるのであれば、そこで分母の2^5が約分されるので分母の数が消えるので、倍積完全数と言うことができるが、この場合はそうではないためここで操作終了となる。

そこで、別の方法を考えた。上の例で見ると、このように操作が進められなくなった際に、「最後に 7 を立てた分母までの計算をリセットし、分母に立てた値（この場合 13）の指数を 2 にして、操作を続ける」ということを行う。つまり、もう一度分母に立てざるを得なくなった素数が出た時、その素数を分母に立てるまでの操作をリセットし、その分母の素数の指数を 1 増やして操作を続ける、ということである。具体的には次のような場合である。

$$\frac{\sigma(n)}{n}=\frac{2\times 3^2}{17}\times\frac{13}{3^2}\times\frac{2\times 7}{13}\times\frac{2^3}{7}\times\frac{3^2\times 7}{2^5}$$

最後の分子に出てきた 3 も 7 もすでに分母に立てられている。そのため、この分子を消すために分母を立てることができない。本来はここで操作が終わってしまうのだが、ここで、最後の分母に立てる値を2^5ではなく、指数を一つ増やし、2^6にすることで、次の式のようになるので、操作を続行することができる。

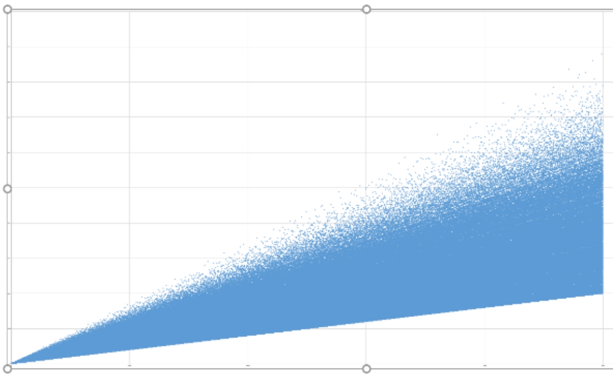
$$\frac{\sigma(n)}{n}=\frac{2\times 3^2}{17}\times\frac{13}{3^2}\times\frac{2\times 7}{13}\times\frac{2^3}{7}\times\frac{127}{2^6}$$

これを行うプログラムも作成したが、１つ目以上の成果は無かった。原因としては、分母の素因数の値が大きくなってしまうため、計算が終わらないことが考えられる。

(Ⅲ)様々な数での約数関数

（ｉ）目的

方法（Ⅱ）でも説明したように我々は Excel を用いて、100 万までの自然数に対して約数の総和とそれが元の数の何倍かを調べた。約数関数$y=\sigma(x)$のグラフをxy平面上に表示すると次のようになる。



(xの変域は 1≦x≦1000000
yの値域は 1≦y≦4390848)

我々は、この約数関数の特徴を探るために、色々な数の約数の総和の特徴を研究した。ここでは、主に$y=\sigma(x!)$に関して述べる

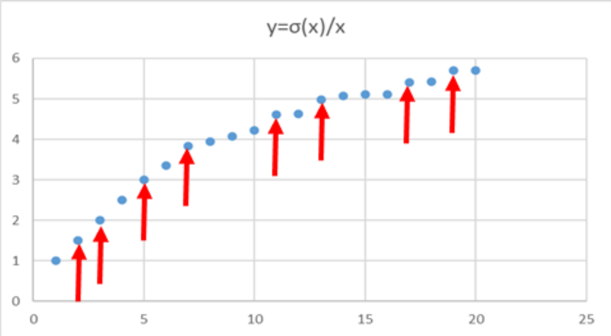
（ⅱ）仮説

我々は、関数のグラフの形や増加の仕方には、何かの特徴があると予想した。

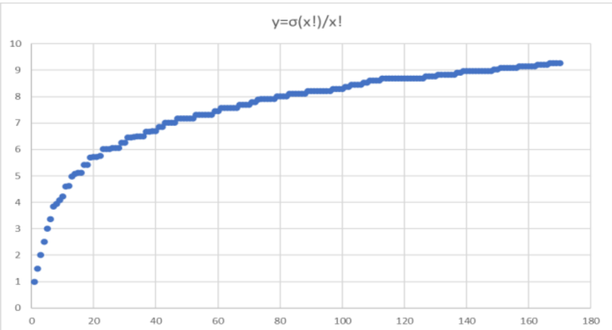
（ⅲ）結果・考察

$y=\sigma(x!)$はxが 1 増えた時のyの増加量が非常に大きく、グラフに示した際に、有効なデータが得られなかったた

め、この研究では$y=\frac{\sigma(x!)}{x!}$ という関数について考えることとした。次がそのグラフである。



上のグラフは、xの変域が $1 \leqq x \leqq 20$ であり、矢印が指している点はxが素数である点である。xが合成数（素数でない数）である点に比べて、yの増加量が大きかった。これを $1 \leqq x \leqq 170$ まで拡張したグラフが下である。



このグラフを見ると、xの値が大きくなるにつれてyの増加量は小さくなっている。

(Ⅳ)約数関数の性質について

（ⅰ）目的

「xに対する約数の総和」の約数の総和、つまり$\sigma(\sigma(x))$を$\sigma^2(x)$と表すこととし、xに対して約数の総和をm回取った時の値を $\sigma^m(x)$と表す。また、$\sigma^2(x)=2x$となるxは「スーパー完全数」と定義されている。我々は$\sigma^m(x)=kx$（kは自然数、以下同様）を満たすxをnとし、$1 \leqq x \leqq 1000000$ の範囲で考えた。

すると、（Ⅱ）（ⅰ）で示したように、$m=1$ の時はそれを満たすxが 9 個だったのに対し、$m=2$ の時は 48 個だった。我々はスーパー完全数などの約数の総和を何度か取ると元の数の自然数倍となる数に着目し、どのような性質や特徴がみられるか研究した。

（ⅱ）仮説

$\sigma^m(n)=kn$の、n,m,kには何らかの関係性がある。

（ⅲ）結果・考察

我々は、まずxをなるべく小さな値に設定して、具体的に何度約数の総和を取れば、xの倍数になるかを調べた。その結果、xが大きな素数になると、$\sigma^m(x)$がxの倍数となる時m,kの値が非常に大きくなっていた。$1 \leqq x \leqq 400$ のxでは、全てのxに対し、m,kが存在した。

$\sigma^m(n)=kn$を満たすm,n,kを $1 \leqq n \leqq 30$ で調べた表が次である。

n	k	m	n	k	m
1	1	1	16	2	2
2	2	2	17	92520	13
3	5	4	18	20	4
4	2	2	19	62720	12
5	24	5	20	84	5
6	2	1	21	3	2
7	24	5	22	49920	13
8	3	2	23	6516224	16
9	168	7	24	7	2
10	12	4	25	881280	17
11	1834560	15	26	28	4
12	10	3	27	3360	9
13	84480	13	28	2	1
14	12	3	29	5.17518E+47	78
15	4	2	30	728	7

6．今後の展望

本研究では、4 つのテーマを扱った。ここではそれぞれのテーマに対して、今後の課題を以下に記す。

（Ⅰ）奇数の完全数について

操作 A という方法を見つけることができたが、具体的に発見するにはまだ${p_1}^{2e_1}\sim{p_n}^{2e_n}$の条件が足りていないと考えたため、条件(A)、(B)以外の条件を探していきたいと思う。

（Ⅱ）倍積完全数について

始めの値の規則や条件を調べることで、試行回数を減らすことができ、より効率的に倍積完全数を発見できると考えたため、その値に関する研究も進めていきたい。

また、2^3と 7 をそれぞれ始めの値として設定したとき、発見された倍積完全数はどちらも 32760 という 4 倍完全数であった。このように異なった始めの値から同じ倍積完全数が得られる現象はどのような条件で起こるのか研究を進めていく。

さらに、操作 A には分母に立てる数の選び方によって多様な分岐がある。同じ初めの値から操作を行ったとしても、分母に立てる値の選び方を変えることで、異なる倍積完全数が発見された。それぞれ、操作の長さも異なるので、選び方によってはより速く見つけることも可能になる。また、選び方によってはkや桁数が大きい倍積完全数を発見できるのではないかと予想している。そこで分母の選び方と、発見される倍積完全数との関係について調べ、より速く見つける方法や、より大きな倍積完全数を発見する方法を探る。

また、2 つ目のプログラムでは、計算が複雑になり時間がかかるため、それを改良し、倍積完全数を発見できなかった始めの値からも、倍積完全数を発見できるようにする。

（Ⅲ）様々な数での倍積完全数

$x!$に対しての約数の総和を取ったが、xを 1 増やしたときのyの増加量はどんどん減っていたので、xを無限に大きくした時yは発散するのか、収束するのかを明らかにしたい。

（Ⅳ）約数関数の性質について

我々は、全ての自然数xは、何回も約数の総和を取れば、xの倍数が出てくるのではないかと考えた。そこで我々は、次のような予想を立てた。

「全ての自然数xは、自然数m,kを用いて、$\sigma^m(x)=kx$と表せる」

今回の研究では、1≦x≦400 のときにこの予想を満たすことを確認できた。xが素数の時は合成数の時に比べてm,kが大きくなる傾向がみられた。また、xが半素数（2 素数の積で表される合成数）の時も大きくなる傾向もあった。xと、m,kの関係や、順番に約数を取っていく以外のm,kを見つける方法も見つけていきたい。

我々は、さらに大きな数に対してこの予想に当てはまるのかその調査も進めていくと同時に、最終的に自分たちの立てた予想を証明する。

７．謝辞

本研究を行うにあたり、学習院大学名誉教授飯高茂先生、本校数学科の吉川先生と加藤先生には大変多くの助言を頂きました。ありがとうございました。

８．参考文献

・完全数の新しい世界Ⅳ　　飯高茂 著
・完全数の新しい世界Ⅵ　　飯高茂 著
・有限と無限の間　　芳沢光雄 著
・素数姫の素数入門　「素数に恋する女」製作委員会 著
・完全数の一覧と性質/高校数学の美しい物語　12/16 最終閲覧 https://manabitimes.jp/math/883
・オイラー双子型メルセンヌ超完全数　梶田光
・完全数の水平展開　飯高茂 http://iitakashigeru.mathacademy.net/kosen2017a.pdf