32201

完全数

3501 石原慶次　3521 鈴木望　3527 永屋和輝　3613 小西光柊

要旨

私たちは、多くの未解決問題に関連している完全数に興味を持ち、それをテーマとして研究した。その中でも、奇数の完全数について考察していく中で、その存在するための条件を知った。それを応用した結果、倍積完全数を効率的に作り出す方法を見つけることができた。また、その方法を用いてプログラムを作成し、大きな倍積完全数を見つけ出すことができた。今後も計算ソフトなどを活用して計算を進め、その方法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。

(例) 6 : 1+2+3+6=12=6×2

→6 は完全数

2022 年 7 月現在では、51 個の完全数が知られている。この完全数に関してその無限性など様々な未解決問題が存在している。

・$\sigma(l)$

自然数lの約数の総和を$\sigma(l)$と表し、これを約数関数と呼ぶ。これには部分的な乗法性があり、互いに素な 2 つの自然数a,bに関して、$\sigma(ab)=\sigma(a)\sigma(b)$である。また、$\sigma(\sigma(l))$を$\sigma^2(l)$と表し、同様に$m$回約数の総和を取るとき$\sigma^m(l)$と表す。

また、この論文では都合上、同じ文字でも異なる箇所で定義しなおして別の文字として用いることがある。

３．使用ソフト

・Excel

・SageMath

４．研究方法

本研究では、主に次の 3 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) 約数関数について

５．研究内容・結果

(Ⅰ)奇数の完全数について

(ⅰ)目的

「２．定義」では、完全数が 51 個発見されていると説明したが奇数のものは 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、その性質を探る研究を行った。

(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。そのため、奇数の完全数にもこ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

まず、全ての自然数Nは素数$p_1,p_2,\dots,p_k,\dots,p_n$と、それぞれの指数$e_1,e_2,\dots,e_k,\dots,e_n$ を用いて

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

と表せる。よって$\frac{\sigma(N)}{N}$は

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

で表せる（以下①とする）。

奇数の完全数（Lとする）を考えるので、${p_k}^{e_k}$は奇素数のみを考えればよい。①の値が 2 になれば完全数であるので、先行研究を参考に、このときに①の分子に 1 つだけ 2 が現れるための条件を示した。

また、分子は偶数になるので、分母の奇素数の累乗のうち一つだけは、約数の総和が、法を 4 として 2 と合同である。よって、その奇素数の累乗の指数は奇数となる。①の中でこれに当たる奇素数をq、その指数をaとする。

$q\equiv 3(mod4)$ の場合、$\sigma(q^a)$ は常に 4 の倍数となるので、適さない。
一方、$q\equiv 1(mod4)$ の場合では、
$a\equiv 1(mod4)$ のとき$\sigma(q^a)\equiv 2(mod4)$
となる。よって、$q\equiv a\equiv 1(mod4)$である。これを条件(A)とする。

式①の分母は奇数、分子は偶数、値は 2 なので、$\sigma(L)\equiv 2L\equiv 2(mod4)$ である。$\sigma(p_k)$奇数である必要があるため、p_kを偶数乗する。これを条件(B)とする。

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及び${p_k}^{2e_k}$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出すことから、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q^aを設定し、分母が全て約分

$$\frac{\sigma(L)}{L}=\frac{2\times 7}{\boxed{13}}\times\frac{3\times 19}{7^2}\times\frac{3\times 127}{19^2}\times\cdots\times\frac{\sigma({p_k}^{2e_k})}{{p_k}^{2e_k}}$$

①　②

によって消えるように${p_k}^{2e_k}$を決定する」という方法を考案した。

① 始めに奇数の累乗（以下、初めの値とする）を条件(A)に従い設定する。右の例では、13^1(図中四角形)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。右図では、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。
そして、次の分数の分母には、 7 と約分して消えるように素数の累乗を立てる。ただし、条件(B)に従って偶数乗である。
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、分母の素因数の積が求める奇数の完全数である。

このような操作（以下操作 A）をすることで、奇数の完全数を発見できるのではないかと考えた。しかし、様々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、${p_k}^{2e_k}$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累乗の約数の総和にある素因数で約分しなければならないことが考えられる。このことから、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、ある数の約数の総和がその自然数倍になる数のことを指す。また、k倍（kは自然数）になる時、その数を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、$1 \leqq n \leqq 1000000$ の自然数nに対して、それが倍積完全数かどうかを調べた。結果は次のようであった。

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

この範囲内で倍積完全数は 9 個存在した。しかし、1000000 個の自然数を調べて、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。そこで我々は、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（以下、効率的に発見する、と表現する）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効率的に発見できるのではないかと考えた。

（ⅲ）結果・考察

①について考える。Nが倍積完全数である場合、①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ことが挙げられるため、それをもとに操作を行った。その一例が次の式である。

$$\frac{\sigma(N)}{N}=\frac{3\times5\times17}{\boxed{2^7}}\times\frac{2\times3^2}{17}\times\frac{2\times3}{5}\times\frac{11^2}{3^4}\times\frac{7\times19}{11^2}\times\frac{2^2\times5}{19}\times\frac{2^3}{7}$$

①　②

$$=5$$

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。（図中四角形）

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

④ 7 を分母に立て、その約数の総和を取った段階で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。

ここで、分母に立てる素因数を分子にある素因数の最大値にしたのだが、それは値が小さい素因数ほど、他の約数の総和の素因数として出てきやすい傾向にあると考えたからである。

また、今回は一つの素数の累乗から計算したが、初めの値が複数の素因数からなる合成数であっても問題はない。

しかし、この方法では初めの値によっては、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、①が異なる素因数ごとに分けて示しているため、$\sigma(l)$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、初めの値を設定することで、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。右がそのプログラムである。

このプログラムを用いて、様々な初めの値を設定して倍積完全数を調べてみた。初めの値は、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、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	出てきた倍積完全数	倍	始めの値	出てきた倍積完全数	倍
2^1	6	2	2^{13}	4.03031×10^{11}	4
2^2	28	2	2^{14}	51001180160	3
2^3	32760	4	2^{16}	8589869056	2
2^4	496	2	2^{18}	1.37439×10^{11}	2
2^6	8128	2	2^{20}	1.80258×10^{21}	5
2^7	14182439040	5	2^{30}	2.30548×10^{18}	2
2^8	459818240	3	2^{60}	2.65846×10^{36}	2
2^9	142990848	4	2^{88}	1.91562×10^{53}	2
2^{10}	1.36619×10^{13}	5	3^6	1.49421×10^{16}	4
2^{11}	5.18667×10^{11}	5	3^{10}	6.08873×10^{15}	4
2^{12}	33550336	2	7^1	32760	4

その結果、上の表にあるような 22 個の倍積完全数を 150 個の数(2 の 1 乗から 100 乗、3 と 5 の 1 乗から 20 乗、7 の 1 乗から 10 乗)を検証した中から発見することができた（紙面の関係上、全ての桁を表すことができていない）。Excel での調査では、100 万までの自然数を全て調べて 9 個しか倍積完全数を発見できなかったため、Excel での調査に比べると、遥かに効率が良いことが分かる。また、見つかった倍積完全数の中で一番大きな数は、初めの値を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。また、この操作ででてくる完全数は、初めの値

```
sage: from typing import List
....: from sympy import factorint
....: 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
....: def prod(x):
....:     y=1
....:     for xn in x:
....:         y*=xn
....:     return y
....: def Func(a):
....:     A=[]
....:     A.extend(a)
....:     return A
....: m=2
....: e=7
....: A=[1]
....: a=[]
....: B=[]
....: b=[]
....: while not(set(A)==set()):
....:     if 1 in A:
....:         A.remove(1)
....:     s=(m^(e+1)-1)/(m-1)
....:     t=factorint(s,multiple=True)
....:     B.extend(t)
....:     b.extend(t)
....:     for i in range(e):
....:         A.append(m)
....:         a.append(m)
....:     A,B=func(A,B)
....:     if set(A) == set():
....:         print("分子：",a)
....:         a,b=func(a,b)
....:         print(b,"倍です")
....:         break
....:     C=Func(A)
....:     c=Func(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)
```

を（メルセンヌ素数+1）にすると作り出せるのだが、これは（Ⅰ）（ⅱ）の式より自明であるため、成果とは言えないと考える。

我々は今回倍積完全数を見つけることができなかった初めの値からも倍積完全数を発見できないかと考え、新たな方法を考案した。倍積完全数を発見できない場合は、具体的には次のような場合である。

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

上の式は、初めの値を2^5として操作を行っている途中であるが、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ため、次に分母に立てる値が無くなってしまったのでここで操作終了となる。このとき、7 を一旦無視してkを 7 の倍数としてもよいのだが、知られているkの最大値は 11 であり、大きくなるにつれて、倍積完全数も減るので、考えない事とした。

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、「最後に分母を立てた以降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、最後に立てた分母以降の計算をリセットし、その素数の指数を 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。

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

最後の分子に出てきた 2 も 7 もすでに分母に立てられている。そのため、次の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 13 ではなく、指数を一つ増やし、13^2にすることで、下のようになるので、操作を続行することができる。

$$\frac{\sigma(n)}{n}=\frac{3^2\times 7}{2^5}\times\frac{2^3}{7}\times\frac{13}{3^2}\times\frac{3\times 61}{13^2}$$

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。

そこで、更にもう 1 パターンの操作を考えた。

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

上の式は2^3を初めの値として操作を行っている途中である。本来ならば次に分母に立てる値は3^2であるが、敢えて3^1を立ててみる。

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

すると、分母が全て消え 3 倍完全数の$2^3\times 3\times 5=120$を作り出すことができた。このように分母に指数を立てるとき、分子にある個数だけ立てるのではなく、1 乗から順に立てていく方法を考えた。

$$\frac{\sigma(n)}{n}=\frac{3\times 5\times 17}{2^7}\times\frac{2\times 3^2}{17}\times\frac{2\times 3}{5}\times\frac{2^2}{3^1}$$

上の例では3^1を立てたが、分母は消えなかった。このまま操作を続けても良いのだが、そうしてしまうと、分子にある3^3はもう分母に立てられなくなってしまう（∵約数関数の乗法性の定義に背くため）。ということは、このnは少なくとも3^3倍完全数とならなければいけなくなるので、作り出せる見込みが減ってしまう。そのため次は3^2を立ててみる。しかし、それでも消えない。3^3の場合でも

同様である。よって、最終的には最初の方法と同様、3^4を立てることとなる。本来分子にあるその素因数の数だけ立てると、一つ目に紹介した方法と同じところに行きつくので、この3つ目の方法は一つ目の方法の拡張と言える。これによって、最初の方法では出てこない倍積完全数を作り出すことができた。

（Ⅲ）約数関数について①

（ⅰ）目的

我々は、約数関数の特徴を探るために、主に二つの約数の総和の特徴を研究した。ここではまず$\sigma^m(l)$について考える。m=2のとき$\sigma^m(l)$がlの2倍になる数、すなわち「lの約数の総和」の約数の総和が$2l$になる数のこと($\sigma^2(l)=2l$)を、「スーパー完全数」という呼び方で定義されている。

我々はm=2のとき$\sigma^m(l)$がlの自然数倍となるlを調べてみた（1≦l≦1000000)。すると、（Ⅱ）（ⅰ）で示したように、m=1の時は、$\sigma^m(l)$がlの自然数倍になるl（倍積完全数)が9個だったのに対し、m=2の時は48個だった。

m=1の時と比べて、m=2の時では$\sigma^2(l)$がlの自然数倍となるlが多く、また$\sigma^2(l)=2l$となるスーパー完全数は7個発見された。そこで、我々はm=3でも同様に$\sigma^m(l)=kl$（kは自然数）となるlの個数を調べてみた。すると、53個の自然数が該当した。このように、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）結果・考察

我々は、まずlをなるべく小さな値を設定して、具体的に何度約数の総和を取れば、lの何倍の数になるかを調べてみた。

l=2のとき、$\sigma(2)=3, \sigma^2(2)=4$より、2は2回約数の総和を取ると2の2倍の4になる。つまり、2はスーパー完全数であることが分かる。

l=3のとき、

$$\sigma(3)=4, \sigma^2(3)=7,$$
$$\sigma^3(3)=8, \sigma^4(3)=15$$

より、3は4回約数の総和を取ると、3の5倍の15となる。

ここで、我々は、全ての自然数l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lは、自然数m, kを用いて、$\sigma^m(l)=kl$と表せる」

まず、この予想について、これを満たす自然数がどれほどあるか実験をしてみた。その結果、1≦l≦400の自然数においてこれが成り立つことが確認された。中にはkが 470 桁となるようなlや、mが550を超えるようなlもあった。しかし、800 まで検証したところ、401を始めとするいくつかの素数において、予想を満たすことを確認することができなかった。この予想を証明することや、それを満たすための条件を考えてみたが、約数の総和という計算でさえ、和と積（素因数分解）が混合しており、一般的なlに対して、2 回目の約数の総和を表すことから難しかった。このように大きな素数になると、m, kの値を出すためには、mの値を非常に大きくする必要があるので、$\sigma^m(l)$や、kの値も大きくなるため、より効率の良い計算方法

が求められる。そこで、lの条件を半素数（2 素数の積からなる合成数）にのみ絞り、以下の証明を与えた。

＜証明＞

2 素数a,bからなる半素数abについて、a,bがともに奇素数であるとき$\sigma(a)\equiv\sigma(b)\equiv 0(mod2)$となるので、$\sigma(a)$と$\sigma(b)$は互いに素でないので

$$\sigma^2(ab)\neq\sigma^2(a)\cdot\sigma^2(b)$$

$\therefore\sigma^2(ab)=\sigma^2(a)\cdot\sigma^2(b)$となるには、$a,b$のどちらか一方が 2 になる必要がある。

a,bは対称性があるので$a=2$とおいても一般性を失わない。

素数bに関して、$\sigma(b)=b+1$より

$b\equiv 1(mod3)$のとき、$\sigma(b)\equiv 2(mod3)$

$b\equiv 2(mod3)$のとき、$\sigma(b)\equiv 3(mod3)$

$b\equiv 0(mod3)$つまり、

$b=3$のとき、$\sigma(b)\equiv 1(mod3)$

$\sigma(a)=3$より、$\sigma(a)$と$\sigma(b)$が互いに素であるとき、2 素数a,bが、$a=2$, $b\equiv 1(mod3)$ または$b=3$ のときである。　　　　　　　証明終

しかし、半素数だけであるので、あまり絞れたとは言えない。他の絞り方も考えたが、今のところいい絞り方は見つかっていない。

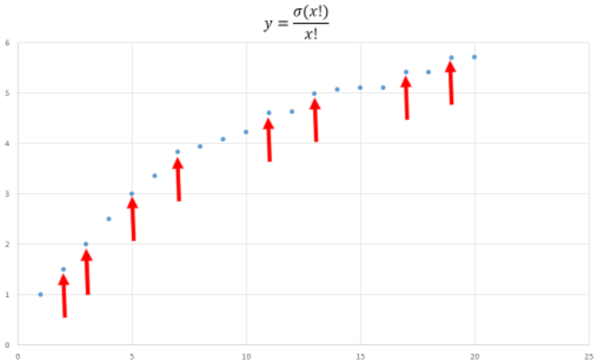
（Ⅲ）約数関数について②

（ⅰ）目的

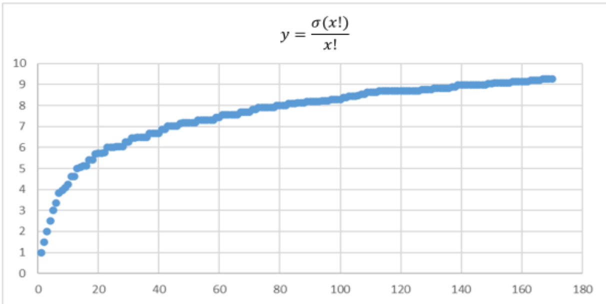
ここでは$y=\sigma(x!)$の研究に関して述べる。

（ⅱ）結果・考察

$\sigma(x!)$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≦x≦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≦x≦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。

6．今後の展望

本研究では、3 つのことをテーマにそれぞれ研究を行った。ここではそれぞれのテーマに対して、今後さらに深めていこうと思っていることをまとめて記す。

（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素因数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

初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を進めていきたい。

また、2^3と7をそれぞれ初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。他にも複数の初めの値でこのような現象が起きた。このように異なった初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大きい方から順ではなく、小さい方から順に立てるように変えたりすることで、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早い段階で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、より早い段階で見つける方法や、より大きな倍積完全数を発見する方法を探る。

また、2つ目の指数を増やす方法では、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初めの値からも、倍積完全数を発見できるようにする。

（Ⅲ）約数関数の性質について

①では自分たちの立てた予想について自然数をしらみつぶしに調べてみたが、見つからなかった場合もあったので、プログラムにおいて最も時間を割かれている素因数分解のアルゴリズムを改善して、今まで確認できなかった素数についても確認できるようにしたい。また、確認できなかったという事は試行回数が多いということなので、どのような自然数においては試行回数が多くなるのかも同時に調べていきたい。

②では、$x!$に対しての約数の総和を取ったが、xを増やしたときの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り、$y=\frac{\sigma(x!)}{x!}$に引いた近似曲線が対数関数であるのはなぜかを調べたりしていく。

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、また、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
・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物語　　2022/06/28 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