# コラッツ予想の第一印象は<br>「3n+1 の式の結果は増大するが 2 のべき乗に収束するだけだよね?」だった!!!

コラッツ予想の証明に取り組むことにしたきっかけを主体にまとめてみました。
※「コラッツ予想」は「コラッツの問題」「3n+1 問題」「3x+1 問題」などとも呼ばれますが「コラッツ予想」に統一します。

## コラッツ予想の式

$$f_{(n)} = \begin{cases} n \div 2 & if\, n \equiv 0 \pmod 2 \\ 3n+1 & if\, n \equiv 1 \pmod 2 \end{cases}$$

コラッツ予想とは、上記の式に対してすべての正の整数(※)のどれかの n で開始し結果を次の f(n)の入力として繰り返しても、最終的に 1 に到達する(正確には 1→4→2→1→…の無限ループになる)と言う予想です。
"2"より"3"が大きいのに、増加し続けずに減少して"1"に到達してしまう。
このように矛盾した結果にたどり着くのは、何故なのでしょうか?
※:「自然数」と言うと'0'を含む場合と含まない場合があるため「正の整数(0 を含まない)」に統一します。

## 第一印象

私のコラッツ予想の式に対する第一印象は「**2 進数との相性がよさそうだな**」と同時に「**3n+1 の式の結果は増大するが 2 のべき乗(※)に収束するだけだよね?**」だった。
※「累乗」とも言いますが「べき乗」に統一します。

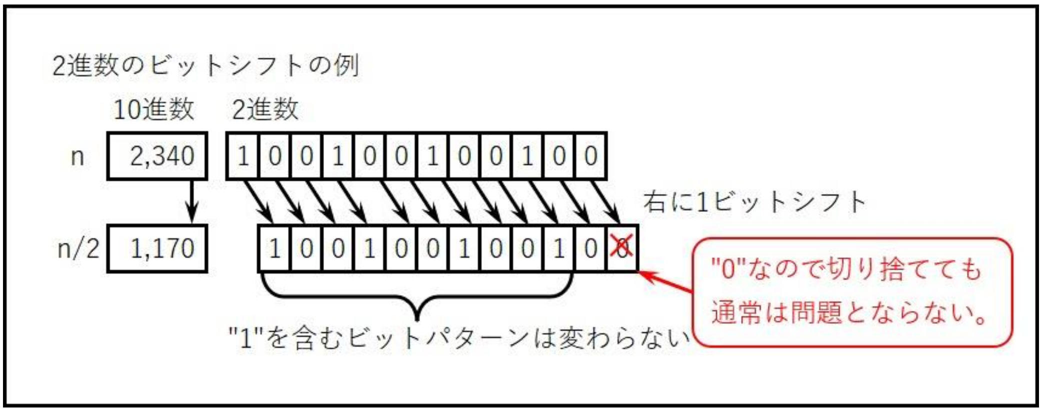
この第一印象は次から来ている

- 偶数時の式 n/2 は 2 進数で見ると右に 1 ビットシフト(右に 1 桁移動)である。
- n=1 に到達した後、1→4→2→1→…の無限ループになるが、出てくる数値はすべて 2 のべき乗である。
- 「すべての正の整数で開始しても n=1 に到達する」のが正しい仮定すると、奇数時の式 3n+1 の結果が 2 のべき乗に収束するだけであろうことが想像できる。

コラッツ予想の式を見て、時間的に 10 秒ぐらいで(まるで呼吸をするかの如く)これを閃いた。
「なぜこんな簡単なことがわからないのだろうか?」と、「コラッツ予想が 80 年以上も証明されてない」ことの方が信じられなかった。
そう、私の論が正しければ、**コラッツ予想の証明そのものは瞬殺だった**と言っても過言ではない。
問題は、この論を**どうやって説明したら伝わるかのみ苦労**しているに過ぎない。

**＜補足＞**

- 右に 1 ビットシフトの例
  偶数の数値を 2 進数で見ると最下位(一番右)の桁は必ず'0'である。
  n/2 はその最下位の'0'を削っていくが、'1'を含む文字列(ビットパターン)に変化は無い。
  これは、確かに桁数は変わっているが数値の要素そのものに変化は無いと言うことです。
  ちなみにプログラミングでは、下位の桁の'0'を削って、限られた演算精度にデータを収めることを簡略化とかモデル化と言い、結果を出力するときに通常は削ったものを元に戻すのですが**コラッツ予想の式にはその仕組みがないため不思議に見える原因の 1 つ**となっている。



・ 2 のべき乗を 2 進数で見ると最上位(一番左)の桁が'1'で下位の桁には'0'が並ぶ。

| 10進数 | 2進数 | 2のべき乗 | 偶数/奇数 |
|---|---|---|---|
| 1 | 1 | $2^0$ | 奇数 |
| 2 | 10 | $2^1$ | 偶数 |
| 4 | 100 | $2^2$ | 偶数 |
| 8 | 1000 | $2^3$ | 偶数 |
| 16 | 10000 | $2^4$ | 偶数 |
| 32 | 100000 | $2^5$ | 偶数 |
| 64 | 1000000 | $2^6$ | 偶数 |
| : | : | : | : |
| 1,024 | 10000000000 | $2^{10}$ | 偶数 |
| : | : | : | : |
| 1,048,576 | 100000000000000000000 | $2^{20}$ | 偶数 |
| : | : | : | : |

このように 2 のべき乗、つまり 1,2,4,8,16,32,64… は数値が大きくなるほど **10 進数で見ると直感的に気が付きにくくコラッツ予想の数列が不思議に見える原因の 1 つ**となっているが、**2 進数で見ると直感的に理解しやすい**。
奇数時の式 3n+1 の結果が 2 のべき乗に収束する(最上位の桁のみに'1')と偶数時の式 n/2 の右に 1 ビットシフト(最下位の桁の'0'を削る)の組み合わせで、結果として n=1 に到達する(その後 1→4→2→1→…の無限ループになる)。
ちなみに、toString(2)などのメソッド(関数)で 2 進数の文字列を取得できるプログラミング言語が多いです。

## 「n=1 になる」と言う表現について

私がコラッツ予想に関して文章を書くときに気を付けているのが**「n=1 になる」と言う表現を使わない**ことです。
**「n=1 に到達した後、1→4→2→1→….のループになる」のが正しい**のであって **n=1 は到達点の一つ**でしかありません。
前者のような**「n=1 になる」という曖昧またはミスリードを誘う表現**をすると目的(ゴール)を見失う可能性が高くなります。
プログラミングの上流工程で、最初にやることの 1 つが「要件定義」です。
要件は、自然言語(日本語など)で書くため、曖昧な表現やミスリードを誘う表現は回避し、より正確な表現にしなければならず、これが言い回しを注意する理由です。

## 第一印象の検証について

「**3n+1 の式の結果は増大するが 2 のべき乗に収束するだけだよね?**」を**証明の結果と仮定**して、検証したものついては下記投稿記事および関連記事を参照ください。
**コラッツ予想のからくりのまとめ**(https://www.seekabypro.com/2021/12/18/collatz_mechanism_summary/)
**iCollatz(無限化コラッツ)の提案 まとめ**(https://www.seekabypro.com/2023/06/21/icollatz_summary/)

また、公開しているツールは簡易確認用の Excel 版と本格確認用の Web ブラウザ(BigInt)版がそれぞれあります。
後者の Web ブラウザ(BigInt)版は、スパコンなどを使うことなく普段ご利用のブラウザで、開始時の n に **10 進数 100 桁や 1,000 桁(無量大数より遥かに大きい、垓(がい)つまり 10 進数 20 桁程度など可愛いもの)などの非常に大きい値**を演算することが可能です(ブラウザの種類やマシン環境(メモリ量など)により制限がある場合があります)。
(注) 投稿記事/ツールは**無料で参照/利用が可能**です。
ただし、**知的財産権(著作権)を放棄したわけではありません**のでご注意ください。

簡単に検証結果をまとめると、3n+1 の「**'3n'の増大が比較的小さく、2 進数で見みたときに'+1'による桁上がり、および桁上がりの連鎖が発生し 2 のべき乗に収束する**」ことを確認しており、仮定が正しい、つまりコラッツ予想の証明ができたと考えています。
なお、「3n の増大が比較的小さく」は、何と比べて小さいのか?
これを明確にし、**コラッツ予想を完全証明**するのが **iCollatz(無限化コラッツ)を提案している理由**です。

## 本文章の知的財産権(著作権)について