第6週の庭いじり

6.1 完全数

6 は興味深い数のひとつだ。それはどういうことかというと、6 の約数は 1, 2, 3, 6 だが、

$$1 + 2 + 3 = 6$$

となっている。この性質を満たす数は 6 の近辺にはない。たとえば 8 は約数として 1, 2, 4, 8 を持つが、明らかに $1 + 2 + 4 \neq 8$ である。約数をもれなく取り出すことができれば、6 の次にこの性質を満たす数が 28 であることを知るのは容易（たやす）い。実際 28 の約数は 1, 2, 4, 7, 14, 28 であるから

$$1 + 2 + 4 + 7 + 14 = 28$$

になっている。この性質を持つ数を**完全数**と呼ぶ。

性質を誤解のないように言えば、完全数とは

自分自身を除く約数の総和が自分自身に等しい数

であると言える。なんと、日本語にする方が式で説明するより難しくなってないかい？　数学とはそんなものだ。

それなら、28 の次の完全数はいくつだろうか。ちょっと計算すればすぐに見つかりそうだ。素数のように、明らかに完全数になり得ないものは省ける。約数の数が少ないものや (素数)^n のような数もたいてい省けそうである。このように候補となる数を絞っていけば、意外に早く見つかるかもしれない。悪いね。そんな簡単には見つからないのだ。よほど根気よくなければ、次の完全数を見つけるのは難しい。根気が勝負になるときこそ **Haskell** の出番というわけだ。

いきなり完全数を求めるスクリプトを書くのも大変だろうから、まずは約数を見つけるスクリプトから見ていこう。

(ghci env.)

```
*Main> let measures n = [a | a <- [1..n], n `mod` a == 0]
*Main> measures 45
[1,3,5,9,15,45]
*Main> measures 911
[1,911]
```

これは対話モードにおいて 1 行で済む。n の約数とは、1 から n までの数のうち、n を割り切ることが出来る数である。それをリスト表記にしただけである。もちろん、素数に対しては 1 と自分自身だけがリスト表示される。

ところで、ここに示した約数を求めるスクリプトは、単純極まりないアルゴリズムであるだけに無駄が多い。当然のことながら、たとえばある数 n が i で割れたなら、その商である $\frac{n}{i}$ も約数である。つまり約数は基本的に 2 つ同時に見つけることができる。基本的と表現したのは、49 が 7 で割れたからと言って、$\frac{49}{7}$ が別の約数とは言えないからだ。しかし、除数 i と商 $\frac{n}{i}$ を一緒にリストに入れてしまえば、割り算は $\sqrt{n}$ まで試せばよいことになる。これは以前、素数を求めた際に使った手法だし、計算量を減らす効果も十分だ。余裕があればこの考え方でスクリプトを書き換えてみればよいだろう。

だが、完全数を求めるためには、これではまったく不十分なのだ。あとで分かるように、完全数は素数とは較べものにならないくらい稀にしか現れない。つまり、計算量が減ること自体は喜ばしいけれど、結局ほとんどの数で調査が不発に終わる。効果的に調べるには、計算量を減らすのではなく、調査しなくてもよい数を飛ばすことである。

さて、約数をすべて列挙するスクリプトが書けたので、それらの和をとれば完全数を探すことができる。

(ghci env.)

```
*Main> let perfectnums pn = [n | n <- [6..pn],   n == sum [a | a <- [1..n-1
], n `mod` a == 0]]
*Main> perfectnums 10000
[6,28,496,8128]
```

スクリプトは関数 measures に sum 関数を追加して、自分自身以外の約数の和を求めている。そのため、変数 n の役割が変わったことに注意してもらいたい。ここで、あとのために言葉の定義をしておきたい。庭いじりの最中だけ通じる用語だが、自分自身以外の約数を “純粋な約数” と呼ぶことにする。よって、6 の “すべての約数” は 1, 2, 3, 6 だが、純粋な約数は 1, 2, 3 である。

その 6 が最初の完全数であることが分かっているので、調査する数は pn = 6 から始めている。与えられた数と純粋な約数の合計が等しくなったら、それが完全数である。

やってみて容易に分かるように、完全数は非常に少ない。5 番目の完全数を探すためには相当の時間を覚悟しなくてはならない。それもそのはずで、完全数は $2^{n-1}(2^n-1)$ の形をしている。いま探した 4 つの数でちょっと確認してほしい。これでは指数関数的に大きな数になってしまうから、完全数がどんなにたくさんあったとしても、気軽に発見できないのだ。

そこで忠告をしておこう。このスクリプトで 5 番目の完全数を探す暴挙に出ないように。5 番目の完全数は 8 桁の数だからといって、**Haskell** の守備範囲だと考えないでほしい。このアルゴリズムは、8128 を見つけるのでさえ少々時間を要する。単純に考えても、1 桁増えるごとに計算時間は 10 倍になるから、8 桁の解を見つけるには 10000 倍の時間がかかる。8128 を 0.1 秒で見つけられても、5 番目の完全数を見つけるには 1000 秒（16 分 40 秒）もかかるのだ。だけど、その間にお茶を入れる時間ができるので、必ずしも悪いことばかりではない。

結局のところ、コンピュータの計算速度を過信してはいけないということだ。そのために、どうしても人の手で、効率的なアルゴリズムが必要になるのである。手っ取り早く 5 番目までの完全数を知りたいなら、$2^{n-1}(2^n-1)$ の形の数だけ調査するスクリプトに変更すればよい。驚くほど早く結果を目にすることができる。でも、今度はお茶を入れる暇がなくなるけど。

試しに $2^{n-1}(2^n-1)$ の形の数が完全数かどうか調べるスクリプトを書いてみよう。と言っても、何の変哲もない恥ずかしいスクリプトだが。

[hs script]
```
ispnum :: Int -> Bool
ispnum n = (2^(n-1)*(2^n-1)) == measuresum (2^(n-1)*(2^n-1))
```

これは単に $2^{n-1}(2^n-1)$ の形の数を `measuresum` 関数に与えただけである。すでに求めた 8128 は、このスクリプトでは $n=7$ の場合にあたる。`measuresum` 関数は愚直に純粋な約数の和を求めていたので、ひどく非効率であったことを思い出してほしい。`ispnum` を使って $n=8$ から順に試してみると、真偽の判断にかかる時間が徐々に長くなることが実感できるだろう。そして、しばらくは `False` が返ってくるばかりである。

(ghci env.)
```
*Main> ispnum 13
True
```

ようやく `True` が返ってくるのが $n=13$ のときで、かなり時間を要したはずだ。しかし $n=13$ のときに完全数になると言われても、このスクリプトは実際の値は示してくれない。みっともない話だが、$2^{13-1}(2^{13}-1)$ を計算させる必要がある。

(ghci env.)

```
*Main> 2^12*(2^13-1)
33550336
```

この程度のことは一度にできるようにして、もっと洗練されたスクリプトにしたいけれど、実行速度が遅いだけでなく measuresum 関数が Int 型でしかないため、調査できる数に上限が存在する。すべきことは $2^{n-1}(2^n-1)$ の形の数について約数を効率的に求めることだ。ただ、それは難しいことではない。

実は、2^n-1 が素数あれば $2^{n-1}(2^n-1)$ は完全数であることが示せる。もし $2^n-1=M$ が素数ならば、$2^{n-1}(2^n-1)=2^{n-1}M$ において純粋な約数、すなわち自分自身を除く約数は

$$1,\ 2,\ 2^2,\ \ldots,\ 2^{n-1},\ M,\ 2M,\ 2^2M,\ \ldots,\ 2^{n-2}M$$

であるから、その和は

$$\begin{aligned} 1+2+2^2+\cdots+2^{n-1}+M(1+2+2^2+\cdots+2^{n-2}) &= \frac{2^n-1}{2-1}+M\cdot\frac{2^{n-1}-1}{2-1} \\ &= \frac{2^n-1}{2-1}+(2^n-1)\cdot\frac{2^{n-1}-1}{2-1} \\ &= (2^n-1)\{1+(2^{n-1}-1)\} \\ &= 2^{n-1}(2^n-1) \end{aligned}$$

となって、もとの数と一致するからである。

実際、$n=7$ のとき $2^7-1=127$ は素数であるから、$2^{7-1}(2^7-1)=8128$ が完全数であることが分かる。

であれば、完全数の探索は 2^n-1 が素数であるかどうかを調べることに他ならない。前に isprime 関数を書いたときは Int 型であったので、Integer 型の isprime' 関数を書けばよい。ただし、単に Int を Integer に変えるだけではだめなので、もうしばらくの庭いじりが必要だ。