

# 計算機科学第一

10 月 20 日

## 今週の目標

- Scala のビルドツール sbt と ScalaTest を用いた単体テストができるようになる。

今日の課題 (提出締切は今週金曜日 23 時 59 分 59 秒)

1. フィボナッチ数を計算する複数のプログラムの作成とそれらを利用したテストをする。
2. パズルを解くプログラムの作成とそのプログラムの単体テストをする。

## 今日のワークフロー

1. GitHub 上の titech-is-cs115/assignment1 を fork する。
2. GitHub 上の #自分のアカウント名#/assignment1 を clone してフィボナッチ数についての課題を終わらせる。
3. GitHub 上の #自分のアカウント名#/assignment1 に push する。
4. GitHub 上の titech-is-cs115/assignment1 に Pull request を送る (次回から課題の提出方法に関するワークフローは書きません)。
5. パズルの課題については...

## 1 フィボナッチ数を計算するアルゴリズムとテスト

### 1.1 概要

プログラムを書くときにまず「効率は悪いけれども確実に正しく動作する実装」をまず書いてから「正しく動作するか少し不安だけど効率がよい実装」を書くことがあるかと思います。今回はフィボナッチ数を例に前者の実装を用いて後者の実装を sbt と ScalaTest を使ってテストしてみましょう。

### 1.2 効率は悪いが確実に正しく動作する実装

フィボナッチ数の定義をそのままプログラムにしまえば確実に正しく動きます。

```
def fib_rec(n: Int) : BigInt = n match {  
  case 0 | 1 => n  
  case _ => fib_rec(n-1) + fib_rec(n-2)  
}
```

ここで BigInt は任意精度整数の型です ( $n$  番目のフィボナッチ数は  $n$  に比べて指数関数的に大きいので単に Int としてしまうと桁溢れます)。

### 1.3 効率はまあまあよいが正しく動作するかちょっと不安な実装

反復を使った  $O(n)$  回の整数演算で  $n$  番目のフィボナッチ数を計算するプログラムです。

---

**Algorithm 1**  $n$  番目のフィボナッチ数を計算するアルゴリズム

---

```
 $m \leftarrow n$   
 $a \leftarrow 1$   
 $b \leftarrow 0$   
while  $m \geq 1$  do  
   $(a, b) \leftarrow (a + b, a)$   
   $m \leftarrow m - 1$   
end while  
return  $b$ 
```

---

### 1.4 効率はかなりよいが正しく動作するか結構不安な実装

行列積を使った  $O(\log n)$  回の整数演算で  $n$  番目のフィボナッチ数を計算するプログラムです。 $n$  番目のフィボナッチ数を  $\text{fib}(n)$  と書くことにします。すると

$$\begin{bmatrix} \text{fib}(n+1) \\ \text{fib}(n) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \text{fib}(n) \\ \text{fib}(n-1) \end{bmatrix}$$

という関係が  $n \geq 1$  について成り立ちます。よって

$$\begin{bmatrix} \text{fib}(n+1) \\ \text{fib}(n) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

が成り立ちます。行列  $A := \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$  とおきます。 $A^n$  を効率よく計算するためにはどのようなすればよいでしょうか。もしも  $n$  が 2 の冪であれば  $A^2 = A \times A$ ,  $A^4 = A^2 \times A^2$ ,  $A^8 = A^4 \times A^4$ , ... と計算していけば  $\log_2 n$  回の行列積の計算で  $A^n$  が計算できます。一般の  $n$  については

$$A^n = \begin{cases} A(A^{(n-1)/2})^2, & \text{if } n \text{ is odd} \\ (A^{n/2})^2, & \text{if } n \text{ is even} \end{cases}$$

という関係を用いればアルゴリズム 2 が得られます。

---

**Algorithm 2**  $A^n$  を計算するアルゴリズム

---

```
 $m \leftarrow n$   
 $r \leftarrow I$   
 $B \leftarrow A$   
while  $m \geq 1$  do  
  if  $m$  is odd then  
     $r \leftarrow r \times B$   
  end if  
   $B \leftarrow B \times B$   
   $m \leftarrow \lfloor m/2 \rfloor$   
end while  
return  $r$ 
```

---

このアルゴリズムは反復回数が  $\lfloor \log_2 n \rfloor + 1$  なので高々  $2(\lfloor \log_2 n \rfloor + 1)$  回の行列積の計算で  $A^n$  が計算できます。このアルゴリズムを用いれば  $O(\log n)$  回の整数演算で  $n$  番目のフィボナッチ数が計算できます。

## 1.5 テストについて

まずフィボナッチ数の計算について定義通りの実装を除く残り 2 つの実装を Scala で書いてください。おそらく for ループなどは計算機科学概論で教えられていないと思うので再帰で書くことになると思います。そして定義通りの実装を使って  $O(n)$  回の整数演算をする実装をテストしてください。定義通りの実装はとても遅いのであまり大きい  $n$  では計算が終わりません。次に  $O(n)$  回の整数演算をする実装を使って  $O(\log n)$  回の整数演算をする実装をテストしてください。ここでは  $n = 100$  くらいでもすぐに計算が終わります。この二段階のテストが上手くいったら自信を持って  $O(\log n)$  回の整数演算をする実装を利用することができます。

## 余談: べき乗に必要な最小の演算回数について

アルゴリズム 2 は行列のべき乗に限らず任意のモノイドにおけるべき乗の計算に適用できます (モノイドとは結合法則を満たす 2 項演算  $\times$  と単位元を持つ集合のことです)。このアルゴリズムは  $\lfloor \log_2 n \rfloor + \text{popcount}(n) - 1$  回のモノイドの演算  $\times$  で  $A^n := \underbrace{A \times A \times \cdots \times A}_{n\text{個}}$  を計算します (最後の  $B \times B$  の演算を除く)。ここで

$\text{popcount}(n)$  は  $n$  の二進数表現に含まれる 1 の数です。しかしこれは必ずしも最小の演算回数ではありません。例えば  $A^{15}$  の計算を考えてみましょう。アルゴリズム 2 では 6 回の演算をします ( $A^8, A^4, A^2$  を計算するのに 3 回、 $A \times A^2 \times A^4 \times A^8$  を計算するのに 3 回)。しかし一方で  $C = A^3$  を 2 回の演算で計算してから  $C^5$  を 3 回の演算で計算すれば 5 回ですみます。関数  $l(n)$  を「 $A^n$  を計算するのに必要な最小の演算回数」と定義します。この関数  $l(n)$  は入力サイズ  $\lfloor \log_2 n \rfloor + 1$  に関して多項式時間で計算できるのでしょうか? そのようなアルゴリズムは未だ知られていません。また直感に反する事実として  $l(2n) = l(n)$  となるような  $n$  の存在が知られています。 $A^{2n}$  は  $A^n$  を二乗すればよいので  $l(2n) = l(n) + 1$  となるような気がするのですがそれは一般には正しくないのです。しかもそれどころか  $l(2n) = l(n) - 1$  となる  $n$  が存在します。これは  $n = 375494703$  のときです。これを実際にプログラムで確認できたらすごいです。 $l(n)$  を計算するプログラムを書いてみるのはいい腕試しになるかもしれません ( $l(2n) = l(n)$  となるような  $n$  は見つけれられます)。

他にも類似の問題として割り算も使える (例えば  $A^7 = A^8/A$  と計算できる) としたときの最小演算回数 ( $\times$  も  $/$  も 1 回と数える) を考えるのも面白いです。実用的には群 (全ての元が逆元を持つモノイド。 $a$  の逆元  $a^{-1}$  は  $a \times a^{-1} = a^{-1} \times a = e$  を満たす元。ここで  $e$  は単位元) において逆元の計算が群演算  $\times$  に比べてずっと効率的な場合、そのような群の上でべき乗を効率的に計算するのに役に立ちます。上記のような性質を持つ群の具体例としては楕円曲線上の群があります (これは説明しません)。