

# sp-17. フィボナッチ数

(Scheme プログラミング)

<https://www.kkaneko.jp/cc/scheme/index.html>

金子邦彦



# アウトライン

17-1 フィボナッチ数

17-2 パソコン演習

17-3 課題



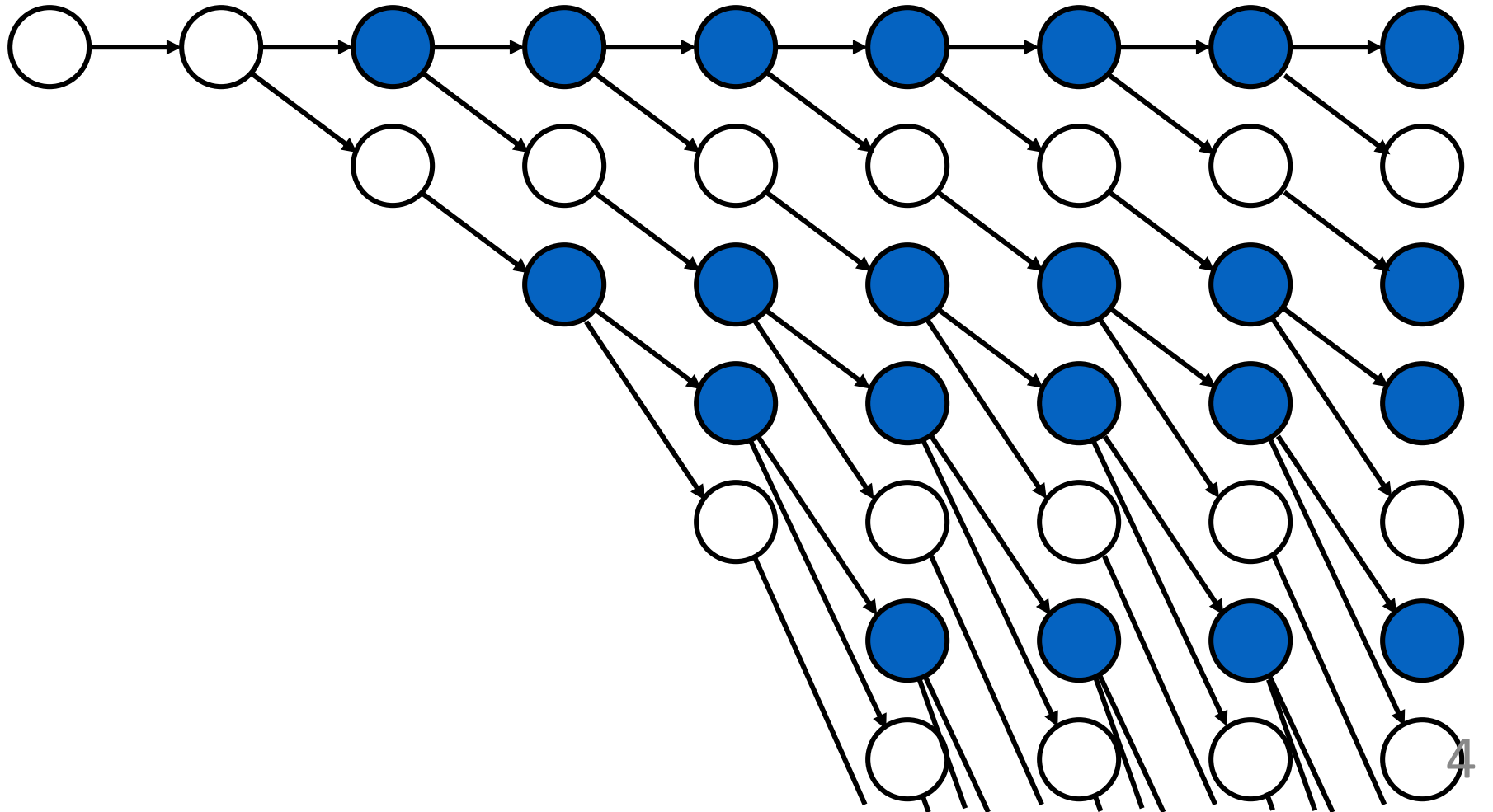


# 17-1 フィボナッチ数

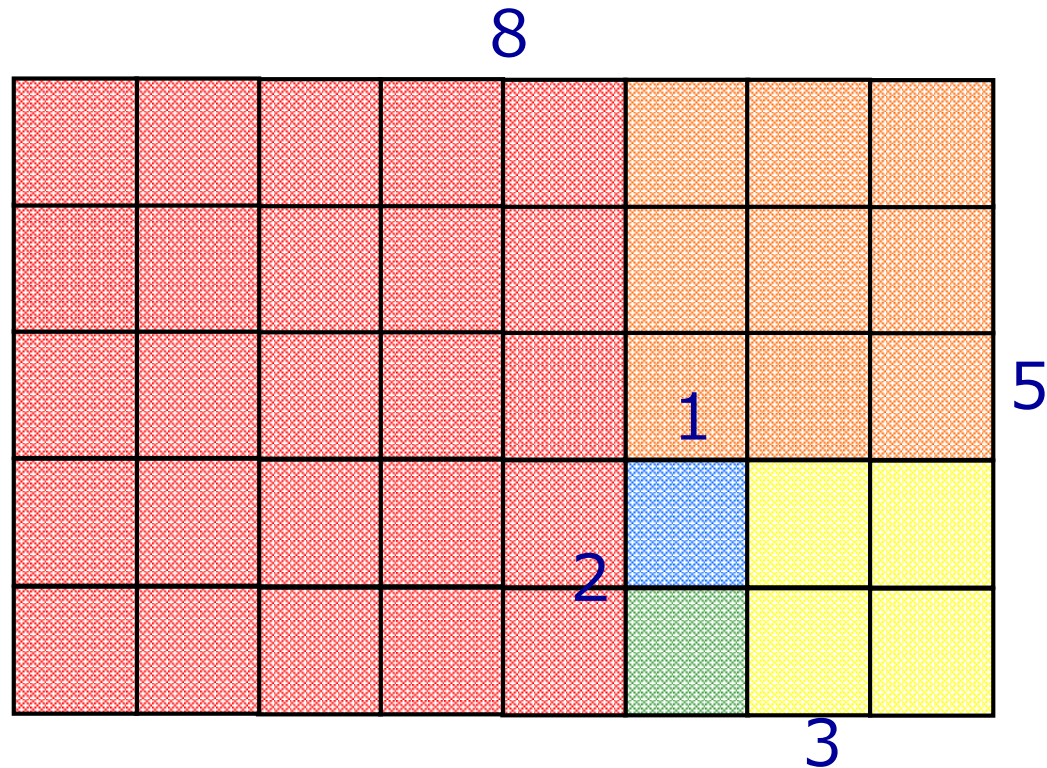
# フィボナッチ数



1      1      2      3      5      8      13      21      34



# フィボナッチ数



# フィボナッチ数



- 生成的再帰（木構造再帰プロセス）の形で、フィボナッチ数

$$f_0 = 0,$$

$$f_1 = 1,$$

$$f_n = f_{n-1} + f_{n-2} (n > 1)$$

の  $i$  番目の数  $f_i$  を計算するプログラムを作る

例) 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ....

# フィボナッチ数



$$f_0 = 0,$$

$$f_1 = 1,$$

$$f_n = f_{n-1} + f_{n-2} \quad (n > 1)$$



# 17-2 パソコン演習



# パソコン演習の進め方



- 資料を見ながら, 「例題」を行ってみる
- 各自, 「課題」に挑戦する
- 自分のペースで先に進んで構いません

# DrScheme の使用



- DrScheme の起動  
プログラム → PLT Scheme → DrScheme
- 今日の演習では「Intermediate Student」  
に設定  
Language  
→ Choose Language  
→ Intermediate Student  
→ Execute ボタン

# 例題 1. フィボナッチ数



- 生成的再帰（木構造再帰プロセス）の形で、フィボナッチ数

$$f_0 = 0,$$

$$f_1 = 1,$$

$$f_n = f_{n-1} + f_{n-2} (n > 1)$$

の  $i$  番目の数  $f_i$  を計算するプログラムを作る

例) 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ....

# 「例題 1. フィボナッチ数」の手順



1. 次を「定義用ウィンドウ」で，実行しなさい
  - 入力した後に，Execute ボタンを押す

```
(define (fibonacci n)
  (cond
    [(= n 0) 0]
    [(= n 1) 1]
    [else (+ (fibonacci (- n 1)) (fibonacci (- n 2)))]))
```

2. その後，次を「実行用ウィンドウ」で実行しなさい

```
(fibonacci 5)
(fibonacci 6)
(fibonacci 7)
```

# 実行結果の例



```
Untitled - DrScheme
File Edit Windows Show Language Scheme Help
Untitled Save Check Syntax Step Execute Break
(define ...)
(define (fibonacci n)
  (cond
    [(= n 0) 0]
    [(= n 1) 1]
    [else (+ (fibonacci (- n 1)) (fibonacci (- n 2)))]))
> (fibonacci 5)
5
> (fibonacci 6)
8
> (fibonacci 7)
13
15:3 Unlocked not running 13
```

# 入力と出力



入力は数値

出力は数値

# フィボナッチ数



```
(define (fibonacci n)
  (cond
    [(= n 0) 0]
    [(= n 1) 1]
    [else (+ (fibonacci (- n 1)) (fibonacci (- n 2)))]))
```

# フィボナッチ数



1.  $n = 0$  ならば :                      → 終了条件  
    0    → 自明な解
2.  $n = 1$  ならば :                      → 終了条件  
    1    → 自明な解
3. そうで無ければ :
  - $(\text{fibonacci}(-n-2))$  と  $(\text{fibonacci}(-n-1))$  を足す  
    ⇒ 結局, fibonacci の実行を繰り返す



# (fibonacci 4)から 3 が得られる過程の概略 (1/2)



$$\begin{aligned}
 (\text{fibonacci } 4) &= (+ (+ (+ \\
 &= \dots & \quad (\text{fibonacci } (-2 \ 1)) \\
 &= (+ (\text{fibonacci } (-4 \ 1)) & \quad (\text{fibonacci } (-2 \ 2))) \\
 & \quad (\text{fibonacci } (-4 \ 2))) & \quad (\text{fibonacci } (-3 \ 2))) \\
 &= \dots & \quad (\text{fibonacci } (-4 \ 2))) \\
 &= (+ (+ & \\
 & \quad (\text{fibonacci } (-3 \ 1)) & \\
 & \quad (\text{fibonacci } (-3 \ 2))) & \quad 1 \\
 & \quad (\text{fibonacci } (-4 \ 2))) & \quad 0) \\
 &= \dots & \quad (\text{fibonacci } (-3 \ 2))) \\
 & & \quad (\text{fibonacci } (-4 \ 2)))
 \end{aligned}$$

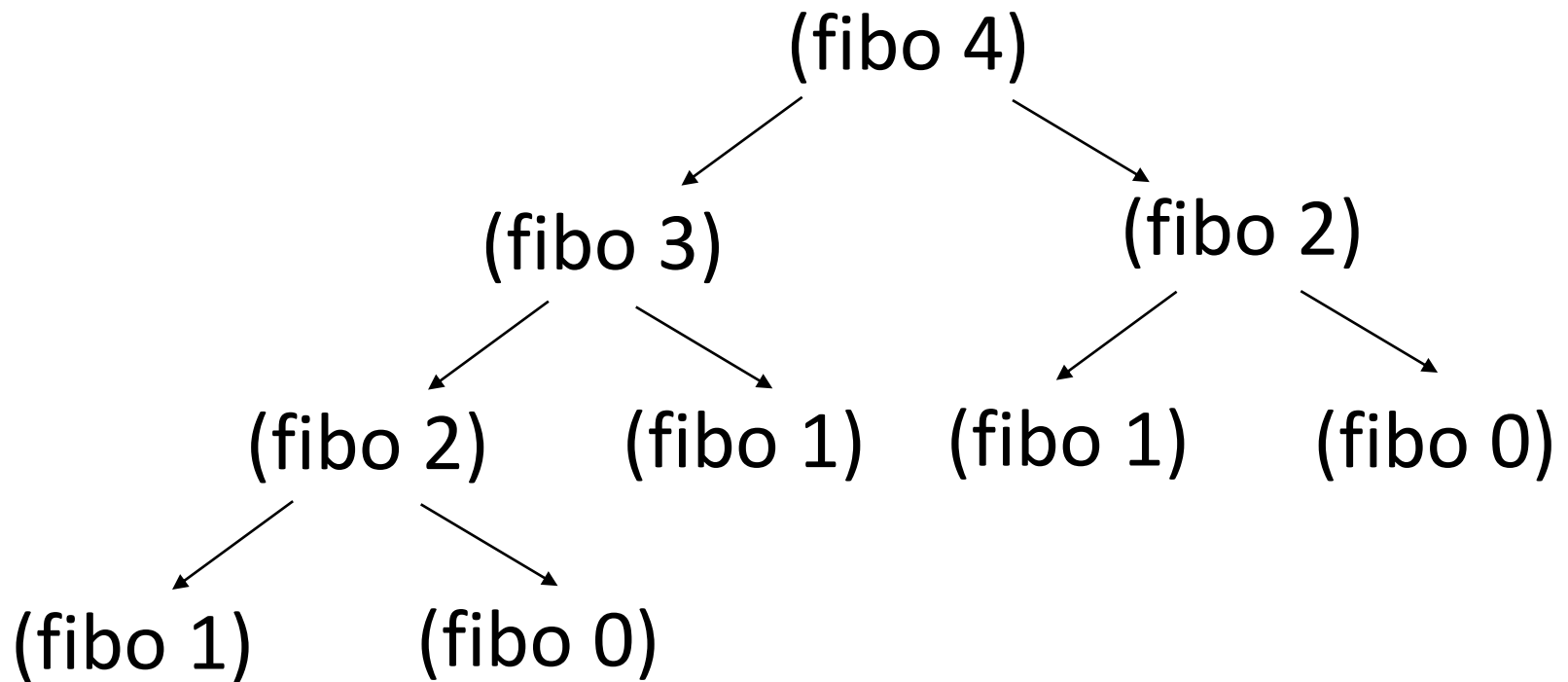
# (fibonacci 4)から 3 が得られる過程の概略 (2/2)



= ...  
= (+ (+  
    1  
    1)  
  (fibonacci (- 4 2))))

= ...  
= (+ 2  
  (+  
    (fibonacci (- 2 1))  
    (fibonacci (- 2 2))))

= ...  
= (+ 2  
  (+ 1 0))  
= ...  
= 3



fibo の計算パターンは、木構造再帰である

# フィボナッチ数



- 関数 fibo は,  $n > 1$  のとき, fibo を2回呼び出す
  - 例) (fibo 10) を計算するために (fibo 9) と (fibo 8) を計算している
- 計算パターンは, 木構造再帰(tree recursion)をなす
  - 樹木状をなす (前ページ)

## 例題 2. 「反復的プロセス」での フィボナッチ数



- フィボナッチ数

$$f_0 = 0,$$

$$f_1 = 1,$$

$$f_n = f_{n-1} + f_{n-2} (n > 1)$$

の  $i$  番目の数  $f_i$  を計算するプログラムを作る

- 例題 1 よりも繰り返し回数が少なくなるように工夫する

## 「例題 2. 反復的プロセスでのフィボナッチ数」の手順



1. 次を「定義用ウィンドウ」で、実行しなさい

- 入力した後に、Execute ボタンを押す

```
(define (fibonacci n)
  (fibonacci-iterate 1 0 n))
(define (fibonacci-iterate a b counter)
  (cond
    [(= counter 0) b]
    [else (fibonacci-iterate (+ a b) a (- counter 1))]))
```

2. その後、次を「実行用ウィンドウ」で実行しなさい

```
(fibonacci 5)
(fibonacci 6)
(fibonacci 7)
```

Untitled - DrScheme

File Edit Windows Show Language Scheme Help

Untitled Save

Check Syntax Step Execute Break

```
(define (fibonacci n)
  (fibonacci-iterate 1 0 n))
(define (fibonacci-iterate a b counter)
  (cond
    [(= counter 0) b]
    [else (fibonacci-iterate (+ a b) a (- counter 1))]))
```

> (fibonacci 5)  
5  
> (fibonacci 6)  
8  
> (fibonacci 7)  
13  
> (fibonacci 8)  
21

実行結果

12:3 Unlocked not running

# 「反復的プロセス」でのフィボナッチ数



```
(define (fibonacci n)
  (fibonacci-iterate 1 0 n))

(define (fibonacci-iterate a b counter)
  (cond
    [(= counter 0) b]
    [else (fibonacci-iterate (+ a b) a (- counter 1))]))
```



# 「反復的プロセス」でのフィボナッチ数



1. まず,  $f_1 (=1)$ ,  $f_0 (=0)$  から始める
2. 次に,  $f_0, f_1$  を使って,  $f_2$  を求める
3. . . .
4.  $n$  に達するまで続ける

# 「反復的プロセス」でのフィボナッチ数



a=1, b=0 から開始して,

$$a \leftarrow a + b$$

$$b \leftarrow a$$

を繰り返す.



終了条件

```
(define (fibonacci-iterate a b counter)
```

```
(cond
```

```
  [(= counter 0) b]
```

```
  [else (fibonacci-iterate (+ a b) a (- counter 1))]))
```

$a \leftarrow a + b$

$b \leftarrow a$

# (fibonacci 4) から 3 が得られる過程の概略



(fibonacci 4)  
= (fibonacci-iterate 1 0 4)  
= ...  
= (fibonacci-iterate 1 1 3)  
= ...  
= (fibonacci-iterate 2 1 2)  
= ...  
= (fibonacci-iterate 3 2 1)  
= ...  
= (fibonacci-iterate 5 3 0)  
= ...  
= 3

a, b, counter の  
値が変化する



## • 例題 1

- 木構造的な再帰
- 計算が冗長

例) (fibonacci 4) の計算では, (fibonacci 2), (fibonacci 1), (fibonacci 0) が繰り返し現れる

## • 例題 2

- 反復的プロセス
- 例題 1 ではあった「冗長な計算」が、例題 2 で無い



# 17-3 課題

# 課題 1



- 例題 1 と例題 2 のフィボナッチ数のプログラムを、性能の面から比較せよ
  - 例題 1 と例題 2 のプログラムについて、(fibonacci 6) を計算するために、例題 1 の fibo, 例題 2 の fibo-iterate が何回繰り返し実行されるか数えよ
  - 例題 1 と例題 2 のプログラムを、DrScheme の stepper で実行し、最終的な結果が得られるまでに「next」ボタンを押した回数（置き換えが起こった回数）を数えてみよ
    - これは、(fibonacci 3), (fibonacci 4), (fibonacci 5), (fibonacci 6) について行え