アルゴリズムとデータ構造入門

1.手続きによる抽象の構築

1.2 プログラムの構築

奥 乃 博
大学院情報学研究科知能情報学専攻
知能メディア講座 音声メディア分野
http://winnie.kuis.kyoto-u.ac.jp/~okuno/Lecture/06/IntroAlgDs/
okuno@i.kyoto-u.ac.jp

TAの居室は10号館4階奥乃1研，2研
TAのページがオープン，質問箱もあります

1

10月24日・本日のメニュー

- 12月5日（火）中間試験（範囲：第1回～第9回）
- 1-1-8 Procedures as Black-Box Abstractions
- 1.2.1 Linear Recursion and Iteration
- 1.2.2 Tree Recursion
- 1.2.3 Orders of Growth
- 1.2.4 Exponentiation
- 1.2.5 Greatest Common Divisors
- 1.2.6 Example: Testing for Primality

2

Ex.1.5

```
(define (p) (p))
(define (test x y)
   (if (= x 0)
       0
       y ))
(test 0 (p))
```

Applicative order	
1	(if (= x 0) 0 (p))
2	(if (= x 0) 0 (p))
3	(if (= x 0) 0 (p))
∞	...

Normal order	
1	(if (= x 0) 0 (p))
2	0

一般にapplicative order で値が求まれば、normal order でも同じ値が求まる

Ex.1.5 のポイント: Special Form

```
(define (p) (p))
(define (test x y)
   (if (= x 0)
       0
       y )))

(define (test-1 x y)
   (and (= x 0) 0 y) )
```

通常のSchemeではどの様な値が返りますか？

```
1.(test 0 (p))

2.(if (= 0 0)
      0
      (p) )

3.(test-1 0 (p))
```

4

Ex.1.6 のポイント: Special Form

```
(define (p) (p))
(define (test x y)
   (if (= x 0)
       0
       y )))
```

通常のSchemeではどの様な値が返りますか？

```
1.(if (= 0 0)
      0
      (p) )

2.(new-if (= 0 0)
      0
      (p) )

(define (new-if pred then-c else-c)
   (cond (pred then-c)
         (else else-c) )))
```

5

10月24日・本日のメニュー

- 1-1-8 Procedures as Black-Box Abstractions
- 1.2.1 Linear Recursion and Iteration
- 1.2.2 Tree Recursion
- **1.2.3 Orders of Growth**
- 1.2.4 Exponentiation
- 1.2.5 Greatest Common Divisors
- 1.2.6 Example: Testing for Primality

10

1-2-1 Linear Recursion and Iteration

- 階乗の定義（その1）

```
(define (factorial n)
   (if (<= n 0)
       1
       (* n (factorial (- n 1))) ))
```

To define N!, *if it is non-positive, return* 1
otherwise, multiply it by (N-1)!

- どう実行されるか。Substition model で実行
- Linear recursive process （線形再帰的プロセス）
  （Nに比例して再帰プロセスが生じる）
- 積は deferred operations（遅延演算）

11

factorial の置換モデルによる実行

Deferred operation

```
(factorial 6)
(* 6 (factorial 5))
(* 6 (* 5 (factorial 4)))
(* 6 (* 5 (* 4 (factorial 3))))
(* 6 (* 5 (* 4 (* 3 (factorial 2)))))
(* 6 (* 5 (* 4 (* 3 (* 2 (factorial 1))))))
(* 6 (* 5 (* 4 (* 3 (* 2 (* 1 (factorial 0)))))))
(* 6 (* 5 (* 4 (* 3 (* 2 (* 1 1))))))
(* 6 (* 5 (* 4 (* 3 (* 2 1)))))
(* 6 (* 5 (* 4 (* 3 2))))
(* 6 (* 5 (* 4 6)))
(* 6 (* 5 24))
(* 6 120)
720
```

12

1-2-1 Linear Recursion and Iteration

- 階乗の定義（その2）

```
(define (factorial n)
  (fact-iter 1 1 n) )

(define (fact-iter product counter max-count)
  (if (> counter max-count)
      product
      (fact-iter (* counter product)
                 (+ counter 1)
                 max-count) ))
```

To define n!, **n! = 1 * 2 * ⋯ * n**
*product = counter * product*
counter = counter + 1

どう実行されるか。Substition model で実行

13

練習問題: 表を埋めてください

手続き	*ステップ数* **各手続きが呼ばれた回数**	*スペース* 遅延演算の個数
(factorial n)		
(fact-iter n)		
テーブル参照型fact		

テーブル参照型というのは，<n, n!> という要素が n の順に並んだもの．

対数表，三角関数の表をひくことを想像すること．

14

factorial の置換モデルによる実行

```
(factorial 6)
(fact-iter   1 1 6)
(fact-iter   1 2 6)
(fact-iter   2 3 6)
(fact-iter   6 4 6)
(fact-iter  24 5 6)
(fact-iter 120 6 6)
(fact-iter 720 7 6)
720
```

- Linear iterative process (線形反復プロセス)

16

1.2.2 Fibonacci – Tree Recursion（木構造再帰）

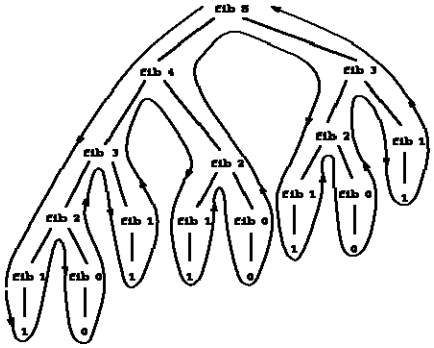
```
(define (fib n)
  (cond ((= n 0) 0)
        ((= n 1) 1)
        (else (+ (fib (- n 1))
                 (fib (- n 2))))))

(define (fib n)
  (fib-iter 1 0 n))

(define (fib-iter a b count)
  (if (= count 0)
      b
      (fib-iter (+ a b) a (- count 1))))
```

17

1.2.2 Fibonacci – Tree Recursion



18

1.2.2 Fibonacci – Iteration

```
(define (fib n)
  (cond ((= n 0) 0)
        ((= n 1) 1)
        (else (+ (fib (- n 1))
                 (fib (- n 2)) ))))
```

- トップダウン(top-down)式に計算

```
(define (fib n)
  (fib-iter 1 0 n))
(define (fib-iter a b count)
  (if (= count 0)
      b
      (fib-iter (+ a b) a (- count 1)) ))
```

- ボトムアップ(bottom-up)式に計算

19

練習問題: 表を埋めてください

手続き	*ステップ数* **各手続きが呼ばれた回数**	*スペース* 遅延演算の個数
(fib n)		
(fib-iter n)		
テーブル参照型 fib		

20

1.2.3 Order of Growth

***R(n)* は、ステップ数あるいはスペース量**

- ***R(n)*** が **Θ(f(n))** $k_1 f(n) \geq R(n) \geq k_2 f(n)$
- ***R(n)*** が **O(f(n))** $R(n) \leq k\ f(n)$
  上限
- ***R(n)*** が **Ω(f(n))** $R(n) \geq k\ f(n)$
  下限

For all $n > n_0$

22

Order of Growth の表を埋めてください

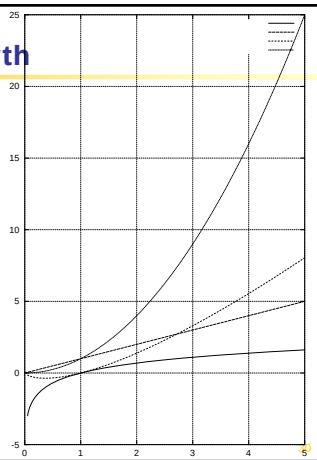
手続き	ステップ数	スペース
factorial		
fact-iter		
テーブル参照型fact		
fib		
fib-iter		
テーブル参照型fib		

23

Order of Growth

次の式はどの曲線・直線に対応するか

- ***y = x***
- ***y = x²***
- ***y = log x***
- ***y = x log x***



24

ギリシャ文字

A	α	alpha	N	ν	nu
B	β	beta	Ξ	ξ	xi
Γ	γ	gamma	O	ο	omicron
Δ	δ	delta	Π	π	pi
E	ε	epsilon	P	ρ	rho
Z	ζ	zeta	Σ	σ	sigma
Ψ	η	eta	T	τ	tau
Θ	θ	theta	Y	υ	upsilon
I	ι	iota	Φ	φ	phi
K	κ	kappa	X	χ	chi
Λ	λ	lambda	Ψ	ψ	psi
M	μ	mu	Ω	ω	omega

31

10月24日・本日のメニュー

- 1-1-8 Procedures as Black-Box Abstractions
- 1.2.1 Linear Recursion and Iteratio
- 1.2.2 Tree Recursion
- 1.2.3 Orders of Growth
- Intermission
- **1.2.4 Exponentiation**
- 1.2.5 Greatest Common Divisors
- 1.2.6 Example: Testing for Primality

12月5日（火）中間試験（範囲：第1回～第9回）

40

1.2.4 Exponentiation（冪乗）

```
(define (expt b n)
  (if (= n 0)
      1
      (* b (expt b (- n 1))) ))
```

$b^n = b * \cdots * b$

- Linear recursive process $\Theta(n)$ steps, $\Theta(n)$ space

```
(define (expt b n)
  (expt-iter b n 1))
(define (expt-iter b counter product)
  (if (= counter 0)
      product
      (expt-iter b
                 (- counter 1)
                 (* b product) )))
```

p =b * p
c = c - 1

- Linear iterative process $\Theta(n)$ steps, $\Theta(1)$ space

41

Exponentiation

```
(define (fast-expt b n)
  (cond ((= n 0) 1)
        ((even? n)
         (square (fast-expt b (/ n 2))))
        (else (* b (fast-expt b
                              (- n 1) )))

(define (even? n)
  (= (remainder n 2) 0) )
```

- recursive process $\Theta(\log n)$ steps, $\Theta(\log n)$ space

42

Exponentiation(べき乗)

- X^{16}
- $16 \equiv 10000_2$ より2進数を4回左シフト

1. まず、1を“SX”、0を“S”で置換し、
2. 次に、先頭の“SX”を除く。
3. 得られたSとXを「Square」「xをかける」と読む。

- 例: X^{23}
- $23 \equiv 10111_2$

1. SX S SX SX SX
2. SSXSXSX
3. X^2 X^4 X^5 X^{10} X^{11} X^{22} X^{23}

43

“Power Tree”

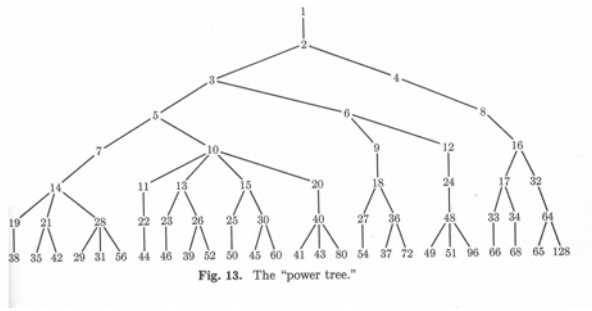


Fig. 13. The “power tree.”

44

10月24日・本日のメニュー

- 1-1-8 Procedures as Black-Box Abstractions
- 1.2.1 Linear Recursion and Iteration
- 1.2.2 Tree Recursion
- 1.2.3 Orders of Growth
- Intermission
- 1.2.4 Exponentiation
- **1.2.5 Greatest Common Divisors**
- 1.2.6 Example: Testing for Primality

45

Greatest Common Divisors（最大公約数）

- *a* mod *b* ＝ *r*（modulo 剰余）とすると
- GCD(*a*, *b*) = GCD(*b*, *r*) が成立。
- ユークリッドの互除法

```
(define (gcd a b)
  (if (= b 0)
      a
      (gcd b (remainder a b)) ))
```

46

Modularity Calculus

- ***a*** ≡ ***b*** mod ***n***（congruent modulo ***n***）
  「***a*** mod ***n*** と ***b*** mod ***n*** が等しい」
- (reminder of) ***x*** modulo ***n*** は剰余
- ***a+b*** mod ***n***
  ≡ ***(a*** mod ***n*** + ***b*** mod ***n)*** mod ***n***
- ***a*b** mod ***n***
  ≡ ***(a*** mod ***n*** * ***b*** mod ***n)*** mod ***n***
- ***a*** mod ***(m*n)***は中国人剰余定理で求める
- $a^{p-1} \equiv 1 \bmod p$ if ***p*** が素数(prime)

47

Chinese Remainder Theorem

連立1次合同式

$x \equiv b_1 \pmod{d_1}$

$x \equiv b_2 \pmod{d_2}$

…

$x \equiv b_t \pmod{d_t}$

の場合、$d_1, d_2, \ldots d_t$ が互いに素であれば、

$n = d_1 d_2 \ldots d_t$

を法として、ただ一つの解がある。

まず、$n/d_i = n_i$ とおけば、d_i と n_i は互いに素であるから、

$n_i x_i \equiv 1 \pmod{d_i}$

の解 x_i を求めることができる。ここで、

$x \equiv b_1 n_1 x_1 + b_2 n_2 x_2 + \ldots + b_t n_t x_t \pmod{n}$

とすれば、この x は明らかにもとの合同式をすべて満足する。

48

Chinese Remainder Theoremの例

x mod *105* は？

- 3 * 5 * 7 = 105
- $x \equiv 1 \pmod 3$
- $x \equiv 2 \pmod 5$
- $x \equiv 3 \pmod 7$
- $35*2 \equiv 1 \pmod 3$
- $21*1 \equiv 1 \pmod 5$
- $15*1 \equiv 1 \pmod 7$ より、
- x mod 105

  $\equiv 1*35*2 + 2*21*1 + 3*15*1 \bmod 105$

  $= 157 \bmod 105 \equiv 52 \bmod 105$

49

Chinese Remainder Theoremの例

2^{90} mod *91* は？

- 91 = 7 * 13
- $2^3 \equiv 1 \pmod 7$ より、$2^{90} \equiv 1 \pmod 7$
- $2^6 \equiv -1 \pmod{13}$ より、

  $2^{84} \equiv 1 \pmod{13} \Rightarrow 2^6 \equiv -1 \pmod{13}$

- $13*6 \equiv 1 \pmod 7$
- $7*2 \equiv 1 \pmod{13}$ より、

- 2^{90} mod *91* $\equiv 1*13*6 - 1*7*2 = 64$

50

宿題：10月30日午後5時締切

12月5日（火）中間試験（範囲：第1回～第9回）

- Tail recursion は iteration に自動変換
- 宿題は、次の6題：
- Ex.1.11, 1.12, 1.13, 1.14, 1.16, 1.19.

51