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

1.3 高階手続きによる抽象化

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

宿題：Ex.1.8

$$\sqrt[3]{x} : y \mapsto \frac{x/y^2 + 2y}{3}$$ で近似

- x は正負両方あることに注意．
- x=-2

1. 初期値 y=1
2. y =(-2/1+2)/3 = 0
3. y =(3/0+0)/3 ⇒ ゼロで割ったエラー

- yがゼロかどうかの判定が必要

10月31日・本日のメニュー

- **1.2.3 Orders of Growth**
- 1.2.4 Exponentiation
- 1.2.5 Greatest Common Divisors
- 1.2.6 Example: Testing for Primality
- 1.3.1 Procedures as Arguments
- 1.3.2 Constructing Procedures Using `Lambda'
- 1.3.3 Procedures as General Methods

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．

4

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		

9

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)) ))
```

```
(fib 6)
(fib-iter  1 0 6)
(fib-iter  1 1 5)
(fib-iter  2 1 4)
(fib-iter  3 2 3)
(fib-iter  5 3 2)
(fib-iter  8 5 1)
(fib-iter 11 8 0)
11
```

- **ボトムアップ(bottom-up)式に計算**

11

1.2.3 Order of Growth Θ(f(n))

R(n) は、ステップ数あるいはスペース量

$\forall n > n_0$, $\exists k$, k_1, k_2, such that

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

通常は上限が使われる

- *R(n)* が Ω(f(n)) 下限 $R(n) \geq k\ f(n)$

14

練習問題：Θを使って表を埋めよ

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		

15

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

25

10月31日・本日のメニュー

- 1.2.3 Orders of Growth
- 1.2.4 Exponentiation
- 1.2.5 Greatest Common Divisors
- **1.2.6 Example: Testing for Primality**
- 1.3.1 Procedures as Arguments
- 1.3.2 Constructing Procedures Using `Lambda'
- 1.3.3 Procedures as General Methods

26

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、
- $x \bmod 105$
  $\equiv 1*35*2 + 2*21*1 + 3*15*1 \bmod 105$
  $= 157 \bmod 105 \equiv 52 \bmod 105$

30

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、
  $2^{84} \equiv 1 \pmod{13} \Rightarrow 2^{90} \equiv -1 \pmod{13}$
- $13*6 \equiv 1 \pmod 7$
- $7*2 \equiv 1 \pmod{13}$ より、
- $2^{90} \bmod 91 \equiv 1*13*6 - 1*7*2 = 64$

31

Lameの定理

- GCD(*a*, *b*)（ただし、*b* < *a*）の計算に*k* step 必要なら、*b* ≥ Fib(*k*)
- 例えば、GCD(*m*, *n*)（ただし、*n* < *m*）が *k* step かかるとすると、*n* ≥ Fib(*k*) ≒ Φ^k / √5

$$\phi = \frac{1}{2}\left(1+\sqrt{5}\right) \qquad Fib(n) \cong \frac{\phi^n}{\sqrt{5}}$$

- つまり、ステップ数は、*n* の対数的に増加。
- Θ(log n) steps

32

10月31日・本日のメニュー

- 1.2.3 Orders of Growth
- 1.2.4 Exponentiation
- 1.2.5 Greatest Common Divisors
- **1.2.6 Example: Testing for Primality**
- 1.3.1 Procedures as Arguments
- 1.3.2 Constructing Procedures Using `Lambda'
- 1.3.3 Procedures as General Methods

34

Testing for Primality（素数の判定）

```
(define (smallest-divisor n)
  (find-divisor n 2))
```

```
(define (find-devisor n test-divisor)
  (cond ((> (square test-divisor) n) n)
        ((divides? test-divisor n) test-divisor)
        (else (find-divisor n
                    (+ test-divisor 1) )) ))
```

```
(define (divides? a b)
  (= (remainder b a) 0) )
```

```
(define (prime? n)
  (= n (smallest-divisor n)) )
```

35

Improvement by HGO

```
(define (smallest-divisor n)
  (find-divisor n 3))
(define (find-devisor n test-divisor)
  (cond ((> (square test-divisor) n) n)
        ((divides? test-divisor n) test-divisor)
        (else (find-divisor n
                  (+ test-divisor 2)) )) )
(define (divides? a b)
  (= (remainder b a) 0) )
(define (prime? n)
  (if (even? n)
      2
      (= n (smallest-divisor n)) ))
```

The Fermat Test

- $a^{p-1} \equiv 1 \bmod p$ if p が素数(prime)

```
(define (expmod base exp m)
  (cond ((= exp 0) 1)
        ((even? exp)
         (remainder (square (expmod base (/ exp 2) m)) m) )
        (else
         (remainder (* base (expmod base (- exp 1) m)) m) )))
```

```
(define (fermat-test n)
  (define (try-it a)
    (= (expmod a n n) a) )
  (try-it (+ 1 (random (- n 1)))))
```

```
(define (fast-prime? n times)
  (cond ((= times 0) true)
        ((fermat-test n) (fast-prime? n (- times 1)))
        (else false) ))
```

37

Probabilistic Algorithms(確率的アルゴリズム)

- Carmichael numbers: 561, 1105, 1072, 2465,
  $a^{560} = (a^2\ a^{10}\ a^{16})^{20}$
  $a^2 \equiv 1 \bmod 3$, $a^{10} \equiv 1 \bmod 11$, $a^{16} \equiv 1 \bmod 17$
  $\Rightarrow\ a^{560} \equiv 1 \bmod 561 = 3 * 11 * 17$
- Fermat's testは、エラーの機会を任意に小さくできる。→ *probabilistic algorithm*
  必要条件のみ満足
- *Algorithm*: Wilson's test
  p is a prime precisely when $(p-1)! \equiv -1 \bmod p$
  必要十分条件

38

Discussion: Fermat's or Wilson's?

1. 単純な素数判定:
2. Fermat's test: p が素数なら
   $\forall a < p,\ a^{(p-1)} \equiv 1 \mod p$
3. Wilson's test: p が素数である必要十分条件は
   $(p-1)! \equiv -1 \mod p$

ちなみに

$n! \sim (2\pi n)^{1/2} (n/e)^n$

39

What is this instrument?

- 計算尺（slide rule, slipstick）
- 対数による積の計算
- 乗算→対数→加算
- 累乗→対数→乗算
- 2^{30} はいくら
- 2^{10} →対数→10log2 →3.01
- $2^{10} \fallingdotseq 10^3$ →1K
- $2^{30} \fallingdotseq 10^9$ →1G
- 音楽のピッチ
- 音の知覚

41

大きな数・小さな数

deca	da	$\times 10^1$	deci	d	$\times 10^{-1}$
hecto	h	$\times 10^2$	centi	c	$\times 10^{-2}$
kilo	K	$\times 10^3$	milli	m	$\times 10^{-3}$
mega	M	$\times 10^6$	micro	μ	$\times 10^{-6}$
giga	G	$\times 10^9$	nano	n	$\times 10^{-9}$
tera	T	$\times 10^{12}$	pico	p	$\times 10^{-12}$
peta	P	$\times 10^{15}$	femto	f	$\times 10^{-15}$
exa	E	$\times 10^{18}$	atto	a	$\times 10^{-18}$
zetta	Z	$\times 10^{21}$	zepto	z	$\times 10^{-21}$
yotta	Y	$\times 10^{24}$	yocto	y	$\times 10^{-24}$

44

10^{1}	ten *or* decad	10^{21}	sextillion
10^{2}	hundred *or* hecatontad	10^{24}	septillion
10^{3}	thousand *or* chiliad		
10^{4}	myriad		
10^{5}	lac *or* lakh		
10^{6}	million	10^{33}	decillion
10^{7}	crore	10^{63}	vigintillion
10^{8}	myriamyriad	10^{303}	centillion
10^{9}	milliard *or* billion	10^{100}	googol
10^{12}	trillion	10^{googol}	googolplex
10^{15}	quadrillion	10^{N}	*N*plex
10^{18}	quintillion	10^{-N}	*N*minex

45

10^{1}	ten *or* decad	10^{21}	sextillion
10^{2}	hundred *or* hecatontad	10^{24}	septillion
10^{3}	thousand *or* chiliad		
10^{4}	myriad	10^{33}	decillion
10^{5}	lac *or* lakh	10^{63}	vigintillion
10^{6}	million	10^{303}	centillion
10^{7}	crore	10^{100}	googol
10^{8}	myriamyriad	10^{googol}	googolplex
10^{9}	milliard *or* billion		
10^{12}	trillion	10^{N}	*N* plex
10^{15}	quadrillion	10^{-N}	*N* minex
10^{18}	quintillion		

46

88plex	無量大数	20plex	垓	4minex	絲 (糸)
80plex	不可思議	16plex	京	5minex	忽
72plex	那由他	12plex	兆	6minex	微
64plex	阿僧祇	8plex	億	7minex	纖 (繊)
56plex	恒河砂	4plex	萬(万)	8minex	沙
48plex	極	3plex	千	9minex	塵
44plex	載	2plex	百	10minex	埃
40plex	正	1plex	十	11minex	渺
36plex	澗	0plex	一	12minex	漠
32plex	溝	1minex	分	13minex	模糊
28plex	穣	2minex	厘	14minex	逡巡
24plex	杼 (禾偏)	3minex	毫(毛)	15minex	須臾

47

1minex	分
2minex	厘
3minex	毫 (毛) モウ
4minex	絲 (糸) シ
5minex	忽 コツ
6minex	微 ビ
7minex	纖 (繊)セン
8minex	沙シャ
9minex	塵ジン
10minex	埃アイ
11minex	渺ビョウ
12minex	漠バク

13minex	模糊
14minex	逡巡 シュンジュン
15minex	須臾シュユ
16minex	瞬息シュンソク
17minex	弾指ダンシ
18minex	殺那
19minex	六徳 リットク
20minex	虚
21minex	空
22minex	清
23minex	浄

48

10月31日・本日のメニュー

- 1.2.3 Orders of Growth
- 1.2.4 Exponentiation
- 1.2.5 Greatest Common Divisors
- 1.2.6 Example: Testing for Primality
- Intermission
- **1.3.1 Procedures as Arguments**
- 1.3.2 Constructing Procedures Using `Lambda'
- 1.3.3 Procedures as General Methods

49

1.3.1 Procedures as Arguments

```
(define (sum-integers a b)
  (if (> a b)
      0
      (+ a (sum-integers (+ a 1) b)) ))
(define (sum-cubes a b)
  (if (> a b)
      0
      (+ (cube a) (sum-cubes (+ a 1) b)) ))
(define (cube x) (* x x x))
(define (pi-sum a b)
  (if (> a b)
      0
      (+ (/ 1.0 (* a (+ a 2))) (pi-sum (+ a 4) b)) ))
```

$$\sum_{i=a}^{b} i$$

$$\sum_{i=a}^{b} i^3$$

$$\sum_{i=a,i+4}^{b} \frac{1}{i(i+2)}$$

```
(define (<name> a b)
  (if (> a b)
      0
      (+ (<term> a)
         (<name> (<next> a) b)) ))
```

50

1.3.1 Procedures as Arguments

```
(define (<name> a b)
  (if (> a b)
      0
      (+ (<term> a)
         (<name> (<next> a) b)) ))
(define (sum term a next b)
  (if (> a b)
      0
      (+ (term a)
         (sum term (next a)next b)) ))
(define (inc n) (+ n 1))
(define (sum-cubes a b)
  (sum cube a inc b) )
(define (identity x) x)
(define (sum-integers a b)
  (sum identity a inc b) )
```

$$\sum_{i=a,next(i)}^{b} f(i)$$

$$\sum_{i=a,i+1}^{b} cube(i)$$

$$\sum_{i=a,i+1}^{b} i$$

51

Pi-Sum (Pi/8) の計算方法

$$\sum_{i=a,\pi next(i)}^{b} \pi term(i)$$

```
(define (pi-sum a b)
  (define (pi-term x)
    (/ 1.0 (* x (+ x 2))) )
  (define (pi-next x)(+ x 4) )
  (sum pi-term a pi-next b) )
```

52

積分(integral) の計算方法

$$\int_a^b f = \left[f\left(a + \frac{dx}{2}\right) + f\left(a + dx + \frac{dx}{2}\right) + f\left(a + 2dx + \frac{dx}{2}\right) + \cdots \right] dx$$

$$\left(\sum_{i=a,i+\Delta x}^{b} f(i) \right) \Delta x$$

```
(define (integral f a b dx)
  (define (add-dx x) (+ x dx))
  (* (sum f (+ a (/ dx 2.0)) add-dx b)
     dx ))
```

53

Ex.1.31 Product

$$\prod_{i=a,next(i)}^{b} f(i)$$

```
(define (product term a next b)
  (if (> a b)
      1
      (* (term a)
         (product term (next a)next b))
  ))
(define (product-cubes a b)
  (product cube a inc b) )

(define (product-integers a b)
  (product identity a inc b) )
```

$$\prod_{i=a,i+1}^{b} i^3$$

$$\prod_{i=a,i+1}^{b} i$$

54

Ex.1.32 Accumulation

$$\sum_{i=a,next(i)}^{b} f(i)$$

```
(define (sum term a next b)
  (if (> a b)
      0
      (+ (term a)
         (sum term (next a) next b)) )
(define (product term a next b)
  (if (> a b)
      1
      (* (term a)
         (product term (next a) next b)) )
```

$$\prod_{i=a,next(i)}^{b} f(i)$$

```
(define (<combiner> <name> <term> a <next> b)
  (if (> a b)
      <null-value>
      (<combiner> (<term> a)
         (<name> <term> (<next> a) <next> b))
  ))
```

Ex.1.32 Accumulation

```
(define (accumulate combiner null-value
   term a next b)
  (if (> a b)
      null-value
      (combiner (term a)
         (accumulate combiner null-value
               term (next a) next b ))))

(define (<combiner> <name> <term> a <next> b)
  (if (> a b)
      <null-value>
      (<combiner> (<term> a)
         (<name> <term> (<next> a) <next> b))
  ))
```

56

10月31日・本日のメニュー

- 1.2.3 Orders of Growth
- 1.2.4 Exponentiation
- 1.2.5 Greatest Common Divisors
- 1.2.6 Example: Testing for Primality
- Intermission
- 1.3.1 Procedures as Arguments
- **1.3.2 Constructing Procedures Using `Lambda'**
- 1.3.3 Procedures as General Methods

57

lambda: Anonymous procedure

```
(define (fact n)
  (if (= n 0)
      1
      (* n (fact (- n 1))) ))
```

は次の式と等価

```
(define fact
  (lambda (n)
    (if (= n 0)
        1
        (* n (fact (- n 1))) )))
```

58

Lambda as anonymous procedure

```
(lambda (x) (+ x 4))
((lambda (x) (+ x 4)) 5)

(define (pi-sum a b)
  (define (pi-term x)
    (/ 1.0 (* x (+ x 2))) )
  (define (pi-next x)(+ x 4) )
  (sum pi-term a pi-next b) )

(define (pi-sum a b)
  (sum (lambda (x) (/ 1.0 (* x (+ x 2))))
    a
    (lambda (x) (+ x 4))
    b ))
```

Using let to create local variables

$$f(x,y) = x(1+xy)^2 + y(1-y) + (1+xy)(1-y)$$

$$a = 1+xy$$
$$b = 1-y$$
$$f(x,y) = xa^2 + yb + ab$$

```
(define (f x y)
  (define (f-helper a b)
    (+ (* x (square a))
       (* y b)
       (* a b) ))
  (f-helper
    (+ 1 (* x y))
    (- 1 y) ))
```

60

1.3.2 Local Variables with `let`

```
(define (f x y)
  (define (f-helper a b)
    (+ (* x (square a))
       (* y b)
       (* a b) ))
  (f-helper
    (+ 1 (* x y))
    (- 1 y) ))
```

```
(define (f x y)
  (let ((a (+ 1 (* x y)))
        (b (- 1 y)) )
    (+ (* x (square a))
       (* y b)
       (* a b) )))
```

```
(define (f x y)
  ((lambda (a b)
     (+ (* x (square a))
        (* y b)
        (* a b) ))
   (+ 1 (* x y))
   (- 1 y) ))
```

```
(let ((<v1> <e1>)
      (<v2> <e2>)
      …
      (<vn> <en>) )
  <body> )
```

シンタックス・シュガー

61

scope of variables

```
(let ((x 7))
  (+ (let ((x 3))
       (+ x (* x 10)) )
     x) )
```

```
(let ((x 5))
  (let ((x 3)
        (y (+ x 2)) )
    (* x y) ))
```

Substition model

62

scope of variables

```
(let ((x 7))
  (+ (let ((x 3))
       (+ x (* x 10)) )
     x) )
```

```
x = 7
  x = 3
    -> 33
x = 7 -> 40
```

```
(let ((x 5))
  (let ((x 3)
        (y (+ x 2)) )
    (* x y) ))
```

```
x = 5
  x = 3
  y = 7
  -> 21
```

63

scope of variables

```
(let ((x 7))
  (+ (let ((x 3))
       (+ x (* x 10)) )
     x)
```

```
x = 7
  x = 3
    -> 33
x = 7 -> 40
```

```
(let ((x 5)
  (let ((x 3)
        (y (+ x 2)) )
    (* x y) ))
```

```
x = 5
  x = 3
  y = 7
  -> 21
```

64

宿題：11月6日午後5時締切

- 不動点
- 宿題は、次の6題：
- Ex.1.21, 1.26, 1.27, 1.30, 1.32, 1.33.

80