第6章 問題の解決

この章のねらい

- モデルの記述（第4章）
- モデル上の操作による計算とプログラム（第5章）
- 本章
  - アルゴリズム：計算手順
    - よいアルゴリズムとよくないアルゴリズムがある
  - 計算のモデル化
    - 有限状態機械，チューリング機械
  - 計算量と計算可能性

アルゴリズムの重要性

- 性能に大きな違いが出る
  - 同じ問題を解く複数のアルゴリズムがある
  - アルゴリズムによって計算時間が桁違いに変わることがある
- 類型化されている
  - 全く違う問題を解くアルゴリズムが同じものになることがある
  - 性能に関する考察・プログラミングを共通化できる

アルゴリズムの実例

- 目的
  - 「アルゴリズム」がどのようなものかを具体的な問題について知る
  - 同じ問題について複数のアルゴリズムを見て、計算時間が変わることを知る
- 紹介される例:

問題	平方根の計算	フィボナッチ数の計算
アルゴリズム	反復法 二分法	再帰法 メモ化法

問題: 平方根の計算

- $\sqrt{x}$ を求める
- 注意:
  - 小数の計算は有限の精度で行われる
  - →近似値しか求められない
- 問題:
  - ある正の実数 x が与えられたときに、2乗すると x に近くなる正の実数 y を精度 d で求める.
  - つまり, $|\sqrt{x} - y| < \delta$ となるような y を 1 つ求める

平方根のアルゴリズム: 反復法

- x = 90, d = 1 の場合を考える
- 「y = 0, 1, 2, 3, ... を順に検討してゆき $(y+d)^2$ が 90 より大きくなったら、その1つ前が解」

アルゴリズム1 (反復による平方根の計算)

$y \leftarrow 0$
while $(y+\delta)^2 < x$ **do**
  $y \leftarrow y + \delta$
done
return y

- 実際の動作 x =2, d =0.0001 の場合、

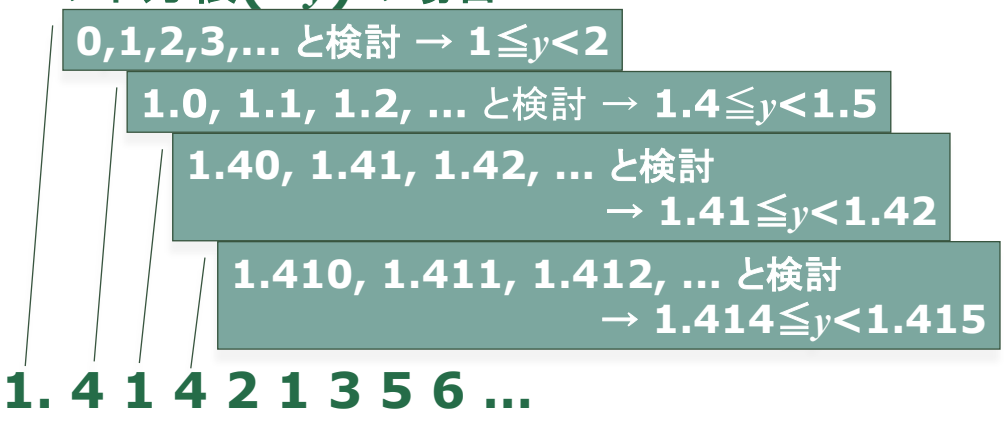
回数	0	1	2	···	14140	14141	14142
候補 (y)	0.0000	0.0001	0.0002	···	1.4140	1.4141	1.4142
$(y+\delta)^2$	0.00000	0.00000	0.00000	···	1.99968	1.99996	2.00024

アルゴリズムの速度

- 繰り返しの回数で比べる
  - プログラムの実行時間の非常におおざっぱな近似
  - 実際のコンピュータの性能と無関係に検討できる
  - 異なる種類の計算の速度差も無視してしまう
- 反復法の場合:
  - 繰り返しの回数は約 $\sqrt{x}/\delta$ 回
  - 精度を1桁増やすと回数も10倍に増える

平方根のアルゴリズム: 二分法の考え方

- アイデア: 1桁ずつ求めてゆく
- 例: 2の平方根(=y)の場合



- 特徴: 解がある範囲を1/10ずつ狭めてゆく
- 単純化: →二分法 (次スライド)

平方根のアルゴリズム: 二分法

アルゴリズム2 (二分法による平方根の計算)
x の平方根を精度 d で求める (ただし $x > 1$):

$a \leftarrow 0$

$b \leftarrow x$

while $b - a > \delta$ **do**

　　$c \leftarrow \frac{a+b}{2}$

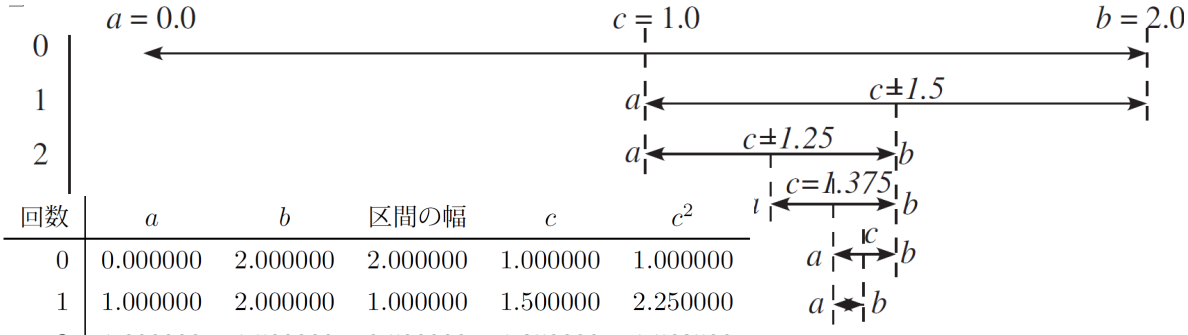
　　if $c^2 > x$ **then** $b \leftarrow c$ **else** $a \leftarrow c$ **endif**

done

return a

平方根のアルゴリズム: 二分法の実際

「区間の幅」が1/2ずつ減ってゆく



回数	a	b	区間の幅	c	c^2
0	0.000000	2.000000	2.000000	1.000000	1.000000
1	1.000000	2.000000	1.000000	1.500000	2.250000
2	1.000000	1.500000	0.500000	1.250000	1.562500
3	1.250000	1.500000	0.250000	1.375000	1.890625
4	1.375000	1.500000	0.125000	1.437500	2.066406
			(中略)		
13	1.414062	1.414307	0.000244	1.414185	1.999918
14	1.414185	1.414307	0.000122	1.414246	2.000091
15	1.414185	1.414246	0.000061		

アルゴリズムの速度

- 反復法: 約 $\sqrt{x}/\delta$ 回
- 二分法:
  - 1回繰り返すごとに区間の幅が1/2になる
  - n回繰り返し後の区間の幅は $x/2^n$
  - これが d以下になるのに要する回数
    → 約 $\log_2 \frac{x}{\delta}$ 回
- 比較: x=2, d=0.0000000001 のとき
  - 反復法: 約141億回
  - 二分法: 35回

小数点以下
10桁まで求める

計算量の考え方

- 重要: 計算時間の見積り
  - 「天気予報」の計算に3日かかっては困る
  - 「百年後に答が出る」は解けないのと同じこと
  - アルゴリズムによって計算時間が大きく違う
- 計算量とは
  - 対象: アルゴリズムの計算時間
    - コンピュータ性能の違いやプログラムの作り方を無視
  - 「問題の大きさ」に対する関係のおおまかな見積り

計算量の使い方

- アルゴリズムどうしを比較をする
  - プログラムを作る前に良いアルゴリズムを選べる
- プログラムの計算時間を予想する
  - 悪いアルゴリズムが現実的な時間で終わらないことが分かる（コンピュータが速くても・工夫をしても1万年以上かかる）
  - 小さな問題の計算時間から大きな問題の時間を予想（x=100のとき100秒 → x=10000のとき ??? 秒）

計算量の見積り方

- 問題の大きさを変数で表わし、
  - 例: n個のデータを処理する
- 計算の回数を式で表わす
  - 例: $3n$+8回, 5 $\log_2(n+1)$回
- 詳細な式ではなく「オーダー」を使う
  - 例: $O(n)$回, $O(\log n)$回
  - ポイント:
    - 定数を無視する
    - 各変数について一番変化の大きい項だけを残す
  - 理由: 定数倍の差はコンピュータの性能の違いやプログラムの作り方ですぐに変わる

計算量の例: 平方根の計算

- 問題: 精度 dで x の平方根を求める
- 問題の大きさ: x と d
- 計算量:
  - 反復法アルゴリズム: $O(\frac{\sqrt{x}}{\delta})$
  - 二分法アルゴリズム $O(\log(\frac{x}{\delta}))$

計算量の違いによる実行時間の差

n	一	十	百	千	万	十万	百万	千万
$\log n$		1ns	2ns	3ns	4ns	5ns	6ns	7ns
n	1ns	10ns	100ns	1μs	10μs	100μs	1ms	10ms
n^2	1ns	100ns	10μs	1ms	0.1s	10s	16分	27時間
n^3	1ns	1μs	1ms	1s	16分	11日	31年	3万年
$2^{n/2}$	1ns	32μs	13日	10^{134}年				

- 1回の計算に1ナノ秒かかる場合の計算時間
- 差が大きい
  → 定数倍の差は無視しても構わない
  - 数倍速くするよりも、計算量の違うアルゴリズムを使う方が良い場合も多い

計算のモデル化

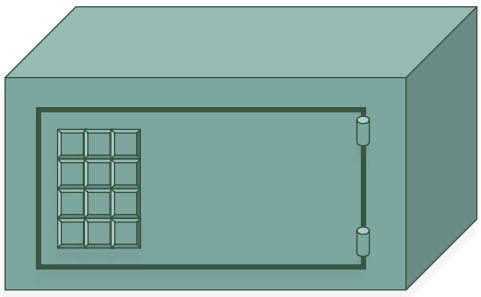
- 計算モデルとは、コンピュータを単純化したもの
  いわば「コンピュータ」の代表
- 計算のモデルを考える意義
  - アルゴリズムの効率を考える
  - 問題がコンピュータで解けるどうかを考える
- モデルの色々
  - 有限状態機械・チューリング機械・ランダムアクセス機械・帰納関数・ラムダ計算・・・

有限状態機械

- 現実のコンピュータで使われる電子回路を単純化したもの
- 機能
  - 信号を入力する(何個かのボタン)
  - 信号を出力する(yes/noだけ)
  - 内部に状態を持つ
- 動き方
  - 入力と現在の状態によって出力と次の状態を決める

有限状態機械の例: 電子錠

- 入力:　ボタン(1,2,3)
- 出力:　錠の開閉
- 働き:　ボタンを「1213」と押すと錠が開く
  - ※「3」と押したときに、過去にどの順でボタンを押したかが大事
  - →内部で覚えておく(状態)



配布不可

有限状態機械の例: 電子錠のモデル化

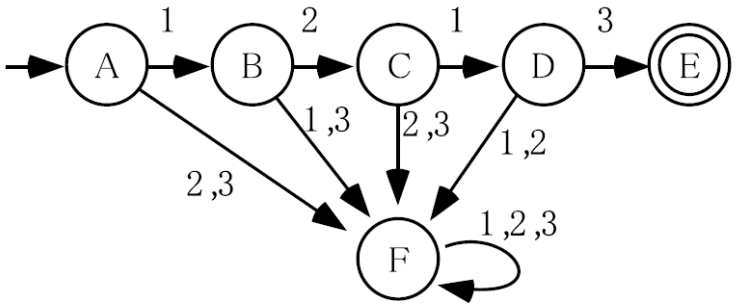
- 状態:
  - A　最初
  - B　「1」と押された
  - C　「12」と押された
  - D　「121」と押された
  - E　「1213」と押された
  - F　押し間違えた
- 出力:
  - 状態がEのときに解錠

- 状態の変化:
  - Aのときに1と押されたらBになる
  - Aのときに2と押されたらFになる
  - Aのときに3と押されたらFになる
  - Bのときに1と押されたらFになる
  - Bのときに2と押されたらCになる
  - Bのときに3と押されたらFになる
  - Cのときに1と押されたらDになる
  - Cのときに2と押されたらFになる
  - Cのときに3と押されたらFになる
  - …

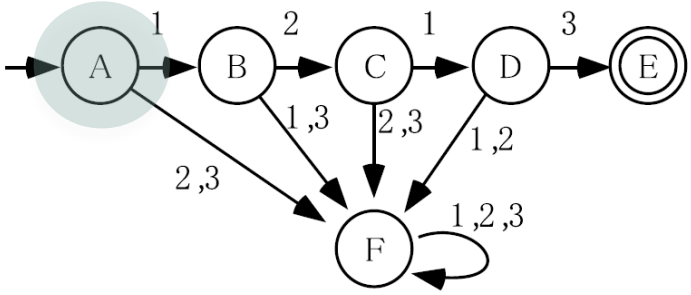
有限状態機械の書き方: 電子錠の場合

現在の状態	A(始)			B			C		
押されたボタン	1	2	3	1	2	3	1	2	3
次の状態	B	F	F	F	C	F	D	F	F

現在の状態	D			E(終)			F		
押されたボタン	1	2	3				1	2	3
次の状態	F	F	E				F	F	F



有限状態機械の働き



状態 ← 最初の状態
while 状態 ≠ 最終状態 **do**
　　ボタン ← 押されたボタン
　　表から (状態, ボタン) に対応する次状態を引く
　　状態 ← 次状態
done

ボタン

1 1
2 2
3 1
1 3
2
1
3

計算のモデル色々

- **有限状態機械**
  - 単純→現実のコンピュータよりも計算能力が低い（記憶装置がない）
- **チューリング機械・ランダムアクセス機械**
  - 有限状態機械 + 記憶装置
  - 現実のコンピュータと「同じ」計算能力
- **帰納関数・ラムダ計算**
  - 数学的な関数を単純化したもの
  - 現実のコンピュータと「同じ」計算能力