アルゴリズムとデータ構造

コンピュータサイエンスコース
知能コンピューティングコース

第1回

塩浦昭義
情報科学研究科 准教授

shioura@dais.is.tohoku.ac.jp
http://www.dais.is.tohoku.ac.jp/~shioura/teaching

この講義について

- 目的：アルゴリズムの解析と設計に必要となる基礎知識の修得
  - アルゴリズムとは？データ構造とは？
  - アルゴリズムをどうやって作るか？
  - アルゴリズムをどのように評価するか？
- 教科書：茨木俊秀「C言語によるアルゴリズムとデータ構造」
- 授業の情報はWebページからも入手可能
- 成績
  - 試験（中間，期末）70%
  - 毎回のレポート30%
    （レポート未提出者は試験の受験不可）
  - 出席は基本的に考慮しません

アルゴリズムとは？

- （計算）問題をコンピュータで解くにはプログラムが必要
- アルゴリズム(algorithm)：
  プログラムの元になる計算手続き
  - ある計算問題が入力として与えられたとき，
    その問題の解（答え）を出力として求めるための，
    機械的操作（計算のステップ）からなる有限の手続き

- アルゴリズムの一例：連立一次方程式の解法

計算問題とは？

- 計算問題(computational problem)
  - 計算により解を求める問題
  - 入力，出力（解）が明確に定義されている
    - 連立方程式の解を求める←計算問題
    - 2つの正整数の最大公約数を求める←計算問題
    - n 個の整数を大きい方から順に並べる←計算問題

    - 地球温暖化の対策を考える←計算問題ではない
    - 人生における幸せとは何か？←計算問題ではない

アルゴリズムの例: 最大公約数を求める

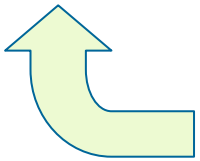
- 与えられた2つの正整数 a_0, a_1 の最大公約数（greatest common divisor, GCD）を求めたい
  ➔ユークリッドの互除法(Euclid's algorithm)
- 例: $a_0 = 315$ と $a_1 = 189$ の最大公約数
- a_2, a_3, … を次のように計算，0 になったら終了
  - a_2 = a_0をa_1で割った余り = 126
  - a_3 = a_1をa_2で割った余り = 63
  - a_4 = a_2をa_3で割った余り = 0

0の直前の値
63が
最大公約数

ユークリッドの互除法

- アルゴリズムを形式的に書いてみる
  - 入力：2つの正整数 a_0, a_1 ($a_0 \geqq a_1 > 0$)
  - 出力：a_0 と a_1 の最大公約数

(1) i := 0 とおく
(2) a_{i+2} :=（a_i を a_{i+1} で割った余り）とおく
(3) a_{i+2} = 0 ならば終了．a_{i+1} を出力
(4) i := i+1 とおき，(2) に戻る



この手順をプログラミング言語を使って厳密・具体的に記述➔プログラム(program)

アルゴリズムの計算量

- a_0 と a_1 の最大公約数を求める単純なアルゴリズム

(1) $b = 1, 2, \ldots, a_0$ に対し，a_0 と a_1 の公約数か否かをチェック

(2) 最大の公約数を出力

ユークリッドの互除法とどちらが良いアルゴリズムか？➔計算時間で比較

- アルゴリズムの時間計算量（時間量，計算時間，time complexity, running time）の数え方：
  問題を解くまでのステップ数を計算
- アルゴリズムの1ステップ：
  加減乗除，余りなどの計算，代入など

時間計算量と入力サイズ

- アルゴリズムの時間計算量
  - 問題の入力サイズ(入力長)の関数として表現
  - 問題の入力サイズ(input size):
    入力データをコンピュータ上で表現したときのサイズ
    ➔ 入力される数値の数，数値のビット長など
  - 例1:a_0 と a_1 の最大公約数
    - 入力サイズは $\log_2 a_0$ と $\log_2 a_1$
  - 例2:n 個の数 a_1, a_2,…, a_n を大きい順に並べる
    - 入力サイズは n もしくは $\log_2 a_1 + \ldots + \log_2 a_n$

単純なアルゴリズムのステップ数

(1) b = 1, 2, …, a_0 に対し, a_0 と a_1 の公約数か否かをチェック

➔ 各 b に対して,

「$a_0 \div b$ の余りは0か？」「$a_1 \div b$ の余りは0か？」をチェック

➔ ステップ数 = $4 \times a_0$

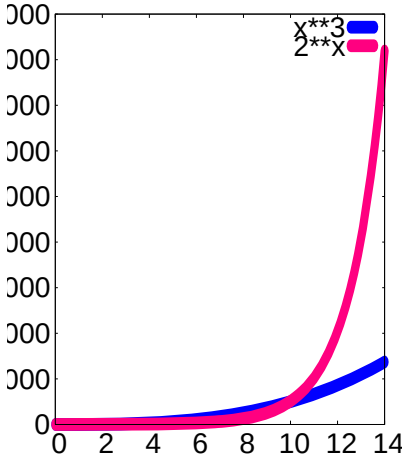
(2) 最大の公約数を出力➔ステップ数 = 1

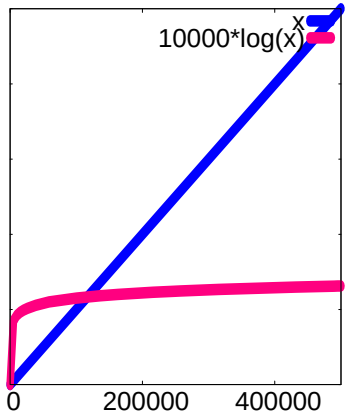
問題の入力サイズ

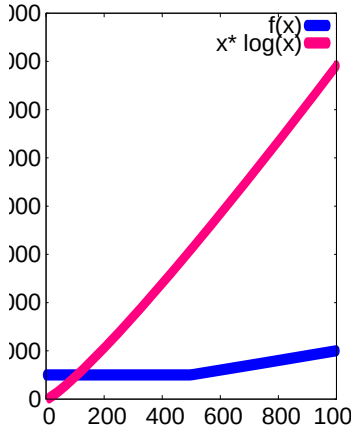
合計ステップ数 = $4a_0 + 1 = 4 \times 2^{\log_2 a_0} + 1$

時間計算量の比較

- アルゴリズムの時間計算量の評価
  - 問題の入力サイズが大きくなったときの
    時間計算量の大小が重要
  - 例1：2^n より n^3 の方がよい
    (n: 問題の入力サイズ)
    $n \geq 10$ ならば $2^n \geq n^3$
  - 例2：n より $10000 \log n$ の方がよい
    係数の大小は関係ない
  - 例3：$n \log n$ より $\max\{500, n\}$ の方がよい
    あくまで n が大きいときの
    大小関係で判断







関数のオーダー記法

- 時間計算量の評価を簡単にするためにオーダー記法(order notation)を使う

ある関数 $T(n)$ と 関数 $f(n)$ に対して $T(n) = O(f(n))$
(「$T(n)$ はオーダー $f(n)$」と読む)
$\Longleftrightarrow$ ある定数 c, n_0 が存在して，$T(n) \leq cf(n)$ $(\forall n \geq n_0)$

– 例1: $2n^2 + 5n + 1000 = O(n^2)$
($n \geq 1000$ のとき，$2n^2 + 5n + 1000 \leq 10n^2$)
n_0 c

– 例2: $\max\{500, n\} = O(n)$
($n \geq 500$ のとき，$\max\{500, n\} \leq n$)

オーダー記法を使う際の注意

- 増加率の高い項のみ考慮
- 係数は無視

例：$2n^2 + 5n + 1000 = O(n^2)$

係数2は無視

5n，1000の項は無視

- オーダー記法での等号は，数学的な意味での等号ではない

例：$2n^2 = O(n^2)$，$2n^2 = O(n^4)$ は正しいが，
$O(n^2) = 2n^2$，$O(n^4) = 2n^2$ は誤り

オーダー記法に関する性質

$T_1(n) = O(f(n))$, $T_2(n) = O(g(n))$ のとき

$T_1(n) + T_2(n) = O(\max\{f(n), g(n)\})$

$T_1(n) \times T_2(n) = O(f(n) \times g(n))$

例：$T_1(n) = O(n^2)$, $T_2(n) = O(n^3)$ のとき

$T_1(n) + T_2(n) = O(n^3)$, $T_1(n) \times T_2(n) = O(n^5)$

オーダー記法 Ω

- T(n)=O(f(n))は，関数f(n)が関数T(n)の上界であることを表す
- 関数の下界を表すための記号は　Ω（オメガ）

ある関数 $T(n)$ と 関数 $f(n)$ に対して $T(n) = \Omega(f(n))$
(「$T(n)$ はオメガ $f(n)$」と読む)
$\Longleftrightarrow$ ある定数 c, n_0 が存在して，$T(n) \geq cf(n)$ $(\forall n \geq n_0)$

– 例：$2n^2 + 5n + 1000 = \Omega(n^2)$
($n \geq 1$ のとき，$2n^2 + 5n + 1000 \geq n^2$)

教科書の定義と少し違います！

$T(n) = \Omega(f(n)) \Longleftrightarrow f(n) = O(T(n))$ が成立

オーダー記法 Θ

- 関数の上界と下界が一致する場合は，Θ（シータ）を使う

ある関数 $T(n)$ と 関数 $f(n)$ に対して $T(n) = \Theta(f(n))$
(「$T(n)$ はシータ $f(n)$」と読む)
$\iff$ ある定数 c_1, c_2, n_0 が存在して，
$c_1 f(n) \leq T(n) \leq c_2 f(n)$ $(\forall n \geq n_0)$

– 例: $2n^2 + 5n + 1000 = \Theta(n^2)$

$T(n) = \Theta(f(n))$
$\iff$ $T(n) = O(f(n))$ かつ $T(n) = \Omega(f(n))$ が成立

時間計算量の評価：最悪と平均

- 時間計算量の（理論的な）評価方法
  - 最悪時間計算量(worst case time complexity)：
    同じ入力サイズの問題例の中で最大の時間計算量を求める
  - 平均時間計算量(average time complexity：
    同じ入力サイズの問題例に対し，それらの時間計算量の平均を求める
  - これ以外の方法もある
- 時間計算量のオーダーが多項式(polynomial) (log n, n^2, n^5,…)
  ➔（理論的には）速いアルゴリズム
  その問題は解きやすい
- 時間計算量のオーダーが指数(exponential) (2^N, N!, N^N,…)
  ➔（理論的には）遅いアルゴリズム
  その問題は解くのが難しい

最大公約数を求める単純なアルゴリズムの時間計算量

(1) b = 1, 2, …, a_0 に対し, a_0 と a_1 の公約数か否かをチェック

(2) 最大の公約数を出力

合計ステップ数 = $4a_0 + 1 = 4 \times 2^{\log_2 a_0} + 1$

最悪時間計算量は $\Theta(2^{\log_2 a_0})$

入力サイズ $\log_2 a_0$ に関して**指数時間**のアルゴリズム

ユークリッドの互除法の時間計算量

- 入力：2つの正整数 a_0, a_1 ($a_0 \geqq a_1 > 0$)
- 出力：a_0 と a_1 の最大公約数

(1) $i := 0$ とおく
(2) $a_{i+2} :=$（a_i を a_{i+1} で割った余り）とおく
(3) $a_{i+2} = 0$ ならば終了．a_{i+1} を出力
(4) $i := i+1$ とおき，(2) に戻る

各ステップは O(1)回の基本演算で実行可能

反復回数の解析：

$a_i > a_{i+1}$, a_{i+2} =（a_i を a_{i+1} で割った余り）$< a_{i+1}$

➔ a_i を a_{i+1} で割ったときの商 $t \geqq 1$,

$a_i = t\, a_{i+1} + a_{i+2} > 2\, a_{i+2}$

➔常に $a_{i+2} < a_i/2$ が成立

ユークリッドの互除法の時間計算量(つづき)

反復回数の解析:

➔常に $a_{i+2} < a_i/2$ が成立

➔$a_{k+1}=0$とすると,

kが偶数のとき: $1 \le a_k < \frac{a_0}{2^{k/2}}$

$$\Rightarrow 2^{k/2} < a_0 \Rightarrow k/2 = \log_2 2^{k/2} < \log_2 a_0$$

$$\Rightarrow k < 2\log_2 a_0$$

kが奇数のとき: $1 \le a_k < \frac{a_1}{2^{(k-1)/2}} < \frac{a_0}{2^{(k-1)/2}}$

$$\Rightarrow 2^{(k-1)/2} < a_0 \Rightarrow (k-1)/2 < \log_2 a_0$$

$$\Rightarrow k < 2\log_2 a_0 + 1$$

➔ 反復回数のオーダーは $O(\log_2 a_0)$

∴ 時間計算量は $O(1) \times O(\log_2 a_0) = O(\log_2 a_0)$

入力サイズ $\log_2 a_0$ に関して多項式時間(線形時間)のアルゴリズム

ユークリッドの互除法の妥当性

- 次の関係を示せばよい:

「a_0とa_1の最大公約数p ＝a_1とa_2の最大公約数q」

- a_0とa_1の最大公約数がp

  ➔ ある正整数 b_0, b_1 が存在して, $a_0=b_0p$, $a_1=b_1p$

- $a_2=a_0$をa_1で割った余り

  ➔ ある正整数 k が存在して $a_2 = a_0-ka_1=(b_0-kb_1)p$

  ➔ pはa_1と a_2の公約数 ➔ p≦q

- a_1とa_2の最大公約数がq

  ➔ ある正整数 c_1, c_2 が存在して, $a_1=c_1q$, $a_2=c_2q$

- $a_0 = ka_1+a_2 = (kc_1+c_2)q$

  ➔ q は a_0 と a_1 の公約数でもある ➔ q≦p

∴ p = q

レポート課題（その１）

問題１：ユークリッドの互除法のアルゴリズムを実現するプログラムを作りなさい．また，それを使って次の2つの数の最大公約数を求めなさい．

a_0＝あなたの生年月日，a_1＝あなたの現住所の郵便番号

（例：1999年5月15日➔a_0＝19990515）

- 提出するもの：作ったプログラム，および最大公約数を求めた際の実行結果
- 提出方法：電子メールにファイルを添付して塩浦のアドレスに送付
  - メールの件名は「アルゴリズムとデータ構造第１回レポート」
  - メール本文に氏名と学籍番号を記入する
  - 大学のメールアカウントより提出する（大学から送信する必要はない）
- 締切：５月６日（木）まで

レポート課題（その2）

問題2：

（1）次のオーダー表記を簡略化せよ

(a) $O(n \log n + n^2) + O(n^{1.83} \log n)$

(b) $O(n^{\log n} + n^{100} + n^{30} \log n)$

(c) $O(n^3 \sin^2 n) \times O(2^n / \log n)$

（2）次の性質を証明せよ

$T_1(n) = O(f(n))$, $T_2(n) = O(g(n))$ のとき

$T_1(n) + T_2(n) = O(\max\{f(n), g(n)\})$

$T_1(n) \times T_2(n) = O(f(n) \times g(n))$

- 提出方法：紙に書いたレポートとして提出
- 締切：4月22日午前9時まで