プログラム設計とアルゴリズム

1. アルゴリズムと計算量

水谷　健太郎
非常勤講師
東京大学　大学院新領域創成科学研究科
特任助教

mizutani-kentaro@aoni.waseda.jp
2019年9月30日

講師略歴

平成8　東京農工大学　工学部電子情報工学科卒業

平成10　東京農工大学　大学院生物システム応用科学研究科修了

平成18　博士（工学）、「機能モジュール統合による自動車運転者の視線推定モデル」（認知脳科学）

現在　東京大学大学院　新領域創成科学研究科
メディカル情報生命専攻　特任助教

国立研究開発法人　産業技術総合研究所
人工知能研究センター　協力研究員

バイオインフォマティクス、ゲノム情報処理関連の高速、大規模計算機技術の研究開発に従事

早稲田大学　非常勤講師「デジタル回路」（春学期）、「Cプログラミング」「プログラム設計とアルゴリズム」（秋学期）担当

本講義について

- 複雑なプログラムを作成する際に必要となる、アルゴリズムとデータ構造について学習します
- プログラミングを含むレポートにより成績をつけます
- プログラミング言語は特に指定しません
  - （が、）Java または C (or C++) のいずれかを使うことを推奨
    ∵電生の1年生でどちらかを必ず学んでいるハズ
    ⇒あまりにもマイナーな言語は要相談！
  - プログラミング言語自体の解説は本講義では、行いません

- 基本的に教科書に沿って講義を行います
  杉原厚吉著、データ構造とアルゴリズム（共立出版）
  （生協に注文済み）

講義予定

- 9月30日　1. アルゴリズムと計算量
- 10月7日　2. リスト構造
- 10月14日　3. ヒープ
- 10月21日　4. ハッシュとバケット
- 10月28日　5. 再帰呼び出しと分割統治
- 11月11日　6. 縮小法
- 11月18日　7. グラフ探索
- 11月25日　8. 最短路問題
- 12月2日　9. 最大流と割り当て問題
- 12月9日　10. 動的計画法
- 12月16日　11. ボロノイ図とドロネー図、3次元凸包
- 12月23日　12. 問題の難しさの測り方
- 1月6日　13. 難問対策
- 1月20日　14. 難問を利用した情報保護
- 1月27日　予備日（休講予定）

「アルゴリズム」とは

問題を解く方法

- ある問題に対して、アルゴリズムが「ある」、「ない」
  例：宝くじに必ず当選する方法？
  - 専門の研究分野「計算可能性理論(computability)」
  - 現実的には実行不可能でも、問題を解く方法が定義できればよい

- アルゴリズムが「ある」なら、良いアルゴリズムとは？
  - 同じ問題なら、速く、簡単に解ける方法が良い
  - もともとの問題が難しい ⇒ 最速のアルゴリズムで解いても時間がかかる

「アルゴリズム」とは

- 与えられたデータから目的の情報を見つけ出したり、作り出したりするための手続きをアルゴリズム（algorithm）という

- 例：
  n個の実数$x_1, x_2, \cdots, x_n$が与えられて、その中の最大値を求めたい．そのための手続き（アルゴリズム）を明確に記述しなさい

最大値を求めるアルゴリズムの記述例1

$x_1, x_2, \cdots, x_n$ を順番に見ていく．その途中ではいつも，「それまでに見た数の中の最大値」を覚えておく．最後まで見終わったとき，「それまでに見た数の中の最大値」が求める最大値である

- ちゃんと手続きが伝わりますか？
- もっと複雑なアルゴリズムをこのような感じで記述していったらどうなるでしょうか？

- 次の記述例と比較してみましょう

最大値を求めるアルゴリズムの記述例2

解くべき問題

入力：正の整数値 n と，n 個の実数値 $x_1, x_2, \cdots, x_n$.　アルゴリズムに与えるデータ

出力：$x_1, x_2, \cdots, x_n$ の中の最大値.　アルゴリズムにより作り出したい情報

手続き：

ステップ 1. $y \leftarrow x_1$;　代入操作

（注：y は「それまでに見た数の中の最大値」を表す.）　コメント

ステップ 2. $i = 2, 3, \cdots, n$ に対して順に次を行う：

$x_i > y$　ならば　$y \leftarrow x_i$;

ステップ 3. y を出力する.

- 入力・出力が明記されている
- 手続きがステップに分割
  - 各ステップで何を行うべきかが明確
  - 変数の使用

アルゴリズムの記述方法に明確なルールがあるのか？

- ない！
  - 流れ図(flow chart)
  - プログラム　Cf. プログラミング言語（Jave, C）
    アルゴリズムをコンピュータが理解できる形で記述したもの

- それぞれの場面や伝えたい相手の知識などに応じて
  - 曖昧性のない厳密さ
  - 必要以上の細部は省略する簡潔さ

  とのバランスを考える必要がある

- バランス感覚に関しては、本講義を通じて（生涯を通して？）身に着ける

最大値を求めるアルゴリズムの記述例3

Algorithm MAX$(x_1, x_2, \cdots, x_n)$: アルゴリズム名（入力変数（引数））

Input: positive integer n, and n real numbers $x_1, x_2, \cdots, x_n$.

Output: maximum of $x_1, x_2, \cdots, x_n$.

Procedure: 手続き

1. $y \leftarrow x_1$;
2. **for** $i \leftarrow 2$ **until** n **do**

   if $x_i > y$ **then** $y \leftarrow x_i$;
3. **return** y.

- プログラミング言語風の記述方法
  ⇒疑似コード(pseudo code)
- 論文では、しばしばこのような記述を見ることが多い

アルゴリズムの優劣

- 同一の目的を達成するアルゴリズムは1つとは限らない

最大値を求めるアルゴリズムの記述例 4

入力：正整数 n と，n 個の実数値 $x_1, x_2, \cdots, x_n$.

出力：$x_1, x_2, \cdots, x_n$ の中の最大値.

手続き：

1. $x_1, x_2, \cdots, x_n$ を小さい順に並べ替えた列 $(y_1, y_2, \cdots, y_n)$ を作る.
2. y_n を出力する.

- 今までのアルゴリズムとは実質的に異なる（どこが？）
- どちらのアルゴリズムがいいアルゴリズムか？
  - 上記のアルゴリズムは「ソート」をしている（最大値を求めるために必ずしも必要ない）
- どのアルゴリズムが優れているかを、評価する方法はあるか？

計算量

- 同一の目的を達成するための複数のアルゴリズムの良し悪しを判断する基準
- プログラムを作成しなくても見積ることができる

- 一般に良し悪しの基準はいろいろと考えられる
  - 計算時間
  - メモリの使用量
  - 通信量
  - 実装のやりやすさ

- まずは計算時間について考えてみる
  ⇒「時間計算量」と呼ぶ

準備：オーダ（ものさし）

- アルゴリズムの速さを、入力の大きさのみの関数として表す
  - 実際にプログラムを作って走らせてみなくても利用可能

- A: あるアルゴリズム
- n: A に与える入力の大きさ（input size）
  - 問題の大きさとも言う

  ⇒このときアルゴリズムが結果を出すために必要な処理時間を$f(n)$と表す
- 一般には$f(n)$の具体的な形はわからない
- しかし、$f(n)$の挙動はさらに簡単な関数$p(n)$で記述できることがある　⇒オーダ

オーダの定義

定義 1.1（オーダ）　$f(n)$ と $p(n)$ を，自然数の上で定義された二つの関数とする．任意の自然数 n に対して

$$\frac{f(n)}{p(n)} < C \tag{1.1}$$

を満たすという性質をもった n によらない定数 C が存在するとき，

$$f(n) = \mathrm{O}(p(n)) \tag{1.2}$$

と書いて，「$f(n)$ は，$p(n)$ のオーダ (order) である」という．

- ビッグ･オー(big-o)記法
- 1.2式を満たすできるだけ簡単な$p(n)$を見つければ、$f(n)$のふるまいを$p(n)$で近似的に眺めることができる

オーダの例

$f(n) = 3n^2 + 8n + 6$ の場合

$p(n) = n^2$ とすると

$$\frac{f(n)}{p(n)} = \frac{3n^2 + 8n + 6}{n^2} = 3 + \frac{8}{n} + \frac{6}{n^2} \leq 3 + 8 + 6 = 17$$

であるから $f(n) = \mathrm{O}(n^2)$

同様に、 $f(n) = \mathrm{O}(2n^2)$ $f(n) = \mathrm{O}(n^3)$ も成り立つ

$p(n) = n\sqrt{n}$ とすると

$\lim n \to \infty$で∞

$$\frac{f(n)}{p(n)} = \frac{3n^2 + 8n + 6}{n\sqrt{n}} = 3\sqrt{n} + \frac{8}{\sqrt{n}} + \frac{6}{n\sqrt{n}}$$

この$p(n)$はオーダの定義を満たさない

よって、最も簡単な $f(n) = \mathrm{O}(n^2)$ を採用する

一般にk次の多項式であれば$\mathrm{O}(n^k)$

時間複雑度(time complexity)

- アルゴリズムAの計算時間$f(n)$に対して
  $f(n)=O(p(n))$と書くことができる場合
  アルゴリズムAの時間複雑度（time complexity）は
  $O(p(n))$であるという

  ⇒アルゴリズムが、最悪の場合でも$p(n)$に比例する計算時間で終了することを表す

- 一般に$f(n)$は、プログラミング言語や計算環境が決まらないと正確にはわからない
- しかし、$f(n)$のオーダはわかる場合がある（次の例）

最大値を求めるアルゴリズム2の時間複雑度

入力：正の整数値 n と，n 個の実数値 $x_1, x_2, \cdots, x_n$.

出力：$x_1, x_2, \cdots, x_n$ の中の最大値.

手続き：

1. $y \leftarrow x_1$;　a 秒、1回

（注：y は「それまでに見た数の中の最大値」を表す.）

2. $i = 2, 3, \cdots, n$ に対して順に次を行う：

$x_i > y$　ならば　$y \leftarrow x_i$;

b 秒、n-1回　　c 秒、最大n-1回

3. y を出力する.

d 秒、1回

a,b,c,dは定数と考えられる
→省略（無視）

$$f(n) = a + (n-1)b + (n-1)c + d = O(n)$$

線形オーダ

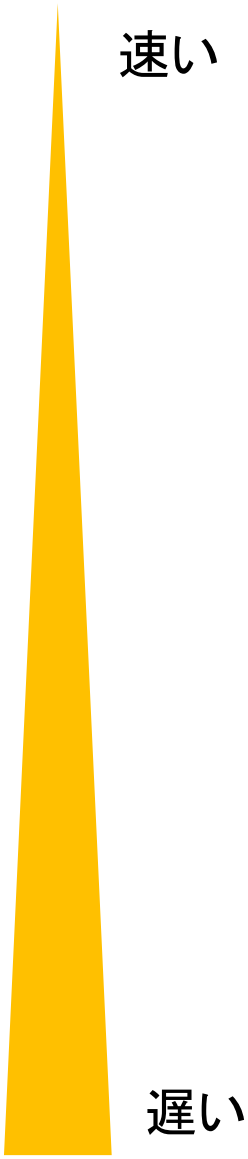
時間複雑度によるアルゴリズムの優劣の比較

- 時間複雑度をもとに、アルゴリズムの優劣を比較することが可能
- 或る2つのアルゴリズムの計算時間が$f(n) = \mathrm{O}(p(n)), g(n) = \mathrm{O}(q(n))$であるとする
  $\mathrm{O}(p(n)) < \mathrm{O}(q(n))$
  なら、$\mathrm{O}(p(n))$より$\mathrm{O}(q(n))$は大きい
  ⇒ nを大きくしていくと、いずれは$f(n) < g(n)$となる
- 「最大値を求めるアルゴリズム4」の時間複雑度は$\mathrm{O}(n \log n)$ であることが知られている
  ⇒ ソートの時間複雑度（第3回で学びます）
- これは、前述のアルゴリズム2の時間複雑度$\mathrm{O}(n)$よりも大きい

  よって、アルゴリズム2の方が時間複雑度の点から優れたアルゴリズムであるといえる

代表的な時間複雑度

- $O(1)$
  定数時間アルゴリズム．最速．nによらない定数．
  （こんなアルゴリズム実現できるのか？）
- $O(\log n)$
  Logオーダアルゴリズム．とても速い
- $O(n)$
  線形時間アルゴリズム．速い
- $O(n \log n)$
- $O(n^2)$
  ビッグデータでは厳しい
- $O(n^3)$
  かなり遅い
- $O(c^n)$ $(c > 1)$
  指数オーダ．遅すぎて実用的ではない



空間複雑度

- 時間と同様に、アルゴリズム中で使われる記憶容量（メモリ）の大きさもオーダで計ることができる

- アルゴリズムAが、大きさnの入力に対して大きさ$g(n)$ビットの記憶容量を必要とし、$g(n)$が

$$g(n) = O(q(n))$$

を満たすとする

- $O(q(n))$を空間複雑度（space complexity）という

- 時間複雑度と空間複雑度をあわせたものを、複雑度（complexity）または計算量という

演習問題（残りの時間）

- 最大値を求めるアルゴリズムを関数（メソッド）として実装しなさい
  - 関数の引数として、任意の数の実数を配列で受け取れるようにする
  - 組み込みの最大値を求める関数（max等）を使うのは禁止
- この関数を使って次のプログラムを作成しなさい
  - コマンドライン引数で「個数」を指定する
  - 引数で指定された個数の実数をランダムに生成して、その中の最大値を出力する
  - プログラムソースには、なるべくコメントを入れること
- 初回なので、プログラムの枠組みを用意しておきました
  - Course N@viからダウンロードしてください
    C用：compute_max.c、Java用：Compute_max.java
  - 実行時間を計測するコードも入っています
- 「個数」をいろいろと変えて計算時間がどのように変わるかを考察しなさい
  - 授業で勉強した「オーダ」との関連性を議論すること
- 以上をレポートにまとめて提出する
  - 提出先：Course N@vi
  - プログラムソース
  - レポート本文はテキストかPDFで添付する
- ✂切：10/5 23:59まで