

アルゴリズムとデータ構造

<http://www-ui.is.s.u-tokyo.ac.jp/~takeo>

五十嵐 健夫

takeo@acm.org

目標

効率のよいプログラムを書くため
基本的な知識・技法を学ぶ

効率＝実行時間とメモリ使用量

コンピュータサイエンスの基礎。

具体的内容

基本的なアルゴリズムとデータ構造を学ぶ

ソート、グラフ、検索、など

計算量の解析について学ぶ

計算量の意味、その計算方法、など

進め方

教科書に沿う

データ構造とアルゴリズム 五十嵐健夫
数理工学社

基本的に教科書が理解できればOK。

成績

毎回出席を取る。

期末テスト

ネット上に過去問あり

スケジュール（仮）

- 10/7 アルゴリズムと計算量 擬似言語、実行時間
基本的なデータ構造 列、スタック、待ち行列、木
- 10/14 集合の表現 ヒープ、2分探索木
- 10/21 集合の表現 平衡木、ハッシュ、集合群
- 10/28 ソート バブルソート、クイックソート
(休講)
- 11/11 ソート マージソート、ヒープソート
- 11/18 ソート バケットソート、基数ソート
- 11/25 有向グラフ ダイクストラ、フロイド、DAG、強成分
(休講)
- 12/2 無向グラフ プリム、クラスカル、関節点、最大マッチング
- 12/16 文字列 KMPアルゴリズム、BMアルゴリズム
- 1/13 設計法 分割統治法、動的計画法、欲張り法、
[テスト] (チェックリスト)
- 1/20

自己紹介

五十嵐 健夫 情報科学科 准教授

専門： ユーザインタフェース
インタラクティブCG

電子ホワイトボード、電子カルテ
3次元モデリング、アニメーション
音声によるインタフェース など

アルゴリズムとは

問題を解く手順

- 1)問題を定式化(モデル化)する
- 2)解法をアルゴリズムとして記述する
- 3)アルゴリズムにしたがって問題を解く

定義:

「明瞭な意味を持ち、有限時間内の有限な
計算で実行できるような命令を有限個並べ
た形で記述される問題の解法」

アルゴリズムとは

段階的詳細化の例(ユークリッドの互除法)

文章で書いたアルゴリズム
擬似言語のプログラム
プログラミング言語による記述

プログラムの実行時間

2つの目標

わかりやすい。構造がシンプルである。
実行時間が早く、メモリを消費しない。

実行時間を決める要素

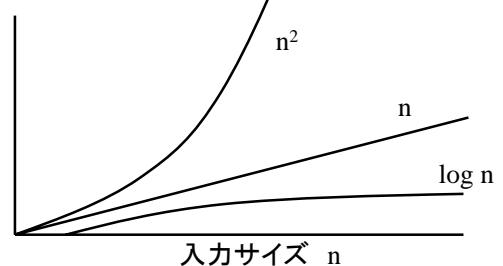
入力データの性質・大きさ
コンパイラの質
ハードウェアの性質
アルゴリズムの計算量

アルゴリズムによって計算時間が変わる例

線形探索と2分探索 n vs $\log_2 n$

計算量の重要性

計算時間



計算量の重要性

n	線形探索	2分探索
1	1ms	1ms
1000	1sec	10ms
1000000	17min	20ms
1000000000	12days	30ms

アルゴリズムの選択が重要(秒-時間-一年)。
ハードウェア・コンパイラ・チューニングなどは小手先。

オーダー記法

「問題のサイズnに対して、予測される計算量の上界値を、定数部分を省略して表現する方法」

正確には、
「アルゴリズムの実行時間が $O(f(n))$ である」とは
「正の定数 c, n_0 が存在して、 n_0 以上のnに対しては
 $T(n) \leq c f(n)$ となる。」
ただし $T(n)$ は大きさnの入力のプログラムの実行時間

(Ω は下界)

オーダー記法

例: 線形探索 $T(n) = an + b$ $O(n)$
2分探索 $T(n) = a \log n + b$... $O(\log n)$

オーダー記法

$O(1) < O(\log n) < O(n^a) < O(n \log n) < O(n^b) < O(\alpha^n) < O(n!)$
 $0 < a \leq 1, 1 < b, \alpha > 0$

プログラムの実行時間

和と積の法則

$$T_1(n) + T_2(n) \dots O(\max(f_1(n), f_2(n)))$$

$$T_1(n) \times T_2(n) \dots O(f_1(n) \times f_2(n))$$

解析は難しいことも多い。

いくつかの規則

一連の文は和の公式 = 最も遅い部分に依存
if文は、長い方に依存
ループは、ループの回数と最長の内部実行時間の積
再帰手続き=再帰方程式を解く

アルゴリズムの選択の注意点

使用回数	多い場合にはオーダーに注意
入力サイズ	大きい場合にはオーダーに注意
保守	保守が必要なら読みやすさ優先
メモリ	外部記憶が使えるか
安定性、精度	数値アルゴリズムで重要

よいプログラミングの習慣

計画的に設計する。段階的詳細化。

オーダーを意識する。

カプセル化・モジュール化する。

既存プログラムを活用する。

汎用性のある・応用の利くコードを書く。

まとめ

講義の進め方

アルゴリズムとは

モデル化と段階的詳細化

計算量の話 オーダー記法