全学共通科目･工学部専門科目

「計算機科学概論」
アルゴリズムとプログラミング
その2

五十嵐 淳
igarashi@kuis.kyoto-u.ac.jp
大学院情報学研究科
通信情報システム専攻

担当分のメニュー

- 6/21, 6/28: アルゴリズムについて
- 7/5: 出張につき休講
- 7/12, 7/19: プログラミングについて
- (補講の予定・内容は未定です)

講義情報

http://www.fos.kuis.kyoto-u.ac.jp/~igarashi/class/cs-intro/

今日のメニュー

- 前回の復習
- ちょっとしたウンチク
- 整列アルゴリズム
- アルゴリズムの正しさ
- 「容易な問題」と「困難な問題」

前回の復習

- アルゴリズムとは:
  - 目的を達成するための停止する手続きの厳密な記述
- 探索アルゴリズム
  - 一列に並べて端から探す線形探索
  - 探索範囲の絞りこみを繰り返す二分探索
- アルゴリズムの時間計算量
  - 入力データの大きさ・長さ(n)に対する最悪・平均
  - n を大きくした時の上限を示す O 記法

ちょっとしたウンチク

- 「アルゴリズム」の語源
  - 8〜9世紀のイスラム数学者: アル・フワーリズミー
    - 著書:「ヒサーブ・アル・ジャブル・ワル・ムカーバラ」
- 記録に残る世界最古(?)のアルゴリズム
  - エウクレイデス(ユークリッド、紀元前3世紀)
  - ふたつの自然数の最大公約数を求める「ユークリッドの互除法」

ユークリッドの互除法

入力: 自然数n, m (ただし n ≧ m)

1. k を n ÷ m の余りとする
2. k について場合分け
   - k = 0 ⇒ m を出力として終了
   - k ≠ 0 ⇒ m, k を入力として互除法を行いその出力をそのまま出力とする

再帰(recursion)

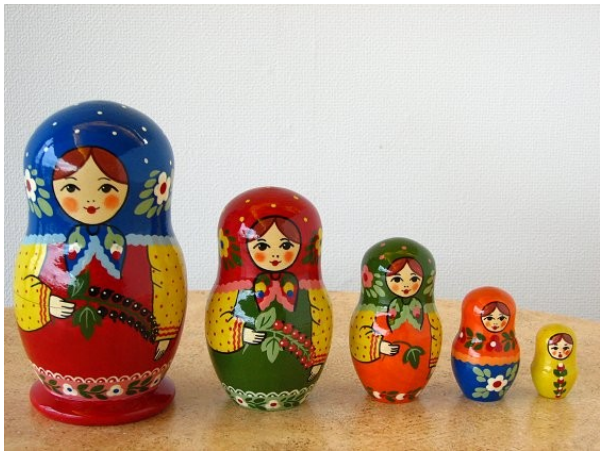
実行例

- n = 1015, m = 455 とする
  - n ÷ m の余りは 105
  - n = 455, m = 105 とする
    - n ÷ m の余りは 35
    - n = 105, m = 35 とする
      - n ÷ m の余りは 0
      - 35を出力
    - 35を出力
  - 35を出力

再帰

何かの中をのぞいたら同じ(ような)ものが出てくる

- 互除法アルゴリズムの中で互除法を使う
- リストとはリストの先頭に要素をひとつ加えたもの
- マトリョーシカ
- 漸化式
- …



- 「始まり」「タネ」がないと意味をなさないので注意

互除法の時間計算量

- 入力の数 m の桁数に比例する回数の割り算でOK
- 桁数 = $O(\log m)$
- 割り算の回数は $O(\log m)$
  - 桁数を入力サイズと考えると $O(N)$

- n と m を素因数分解するより高速
  - 素因数分解は「困難な問題」(と考えられている)

今日のメニュー

- 前回の復習
- ちょっとしたウンチク
- 整列アルゴリズム
- アルゴリズムの正しさ
- 「容易な問題」と「困難な問題」

整列(sorting)問題

- 入力 X: データの集まり(サイズn)
  - データには順序(全順序)がついている
  - 単純のためデータは相異なるとする
- 出力 Y: 入力データを順序に従って並びかえたもの
  - 配列の場合
    $\forall i$ s.t. $0 \leqq i < N-1$, $Y[i] \leqq Y[i+1]$
    $\forall i$ s.t. $0 \leqq i < N-1$, $\exists$ j, $X[i] = Y[j]$

整列アルゴリズム

- 素朴な $O(n^2)$ アルゴリズム
  - うましかソート
  - バブルソート(bubble sort)
  - 選択ソート(selection sort)
  - 挿入ソート(insertion sort)
- 高速な $O(n \cdot \log n)$ アルゴリズム
  - マージソート (merge sort)
  - クイックソート (quick sort)
  - ヒープソート (heap sort)

うましかソート

- i = 0, 1, ..., n-2 まで以下を繰り返す
  - j = i+1, i+2, ..., n-1まで以下を繰り返す
    - X[i] と X[j] を比較し、X[j] が小さかったら X[i] と交換
- X を出力

- i = 0 の時、j についての繰り返し終了時には X[0] に X の最小値が入る
- i = 1 の時、j についての繰り返し終了時 には X[1] に X の二番目に小さい値が入る

うましかソートの時間計算量

- 大小比較の回数は(データに依らず)
  $(n-1) + (n-2) + \ldots + 1 = n(n-1)/2 = O(n^2)$
- 交換の回数は $0 \sim n(n-1)/2 = O(n^2)$

- 整列アルゴリズムでは特に比較の回数に注目する

マージソート

基本アイデア:

- 整列済のふたつの列から、それらを併合(マージ)した整列済の列を得るのは簡単
- 問題サイズを半分半分にする

問題を簡単に解けるくらい
小さく分割してから、その
結果をうまく合成して全体の
結果を得る

マージ

- 入力: 整列済の列 X, Y

X

11	19	25	48	61	...			

Y

3	8	14	26	70	...			

- 出力: X, Y の要素を過不足なく含む整列済の列 Z

Z

マージのアルゴリズム(配列版)

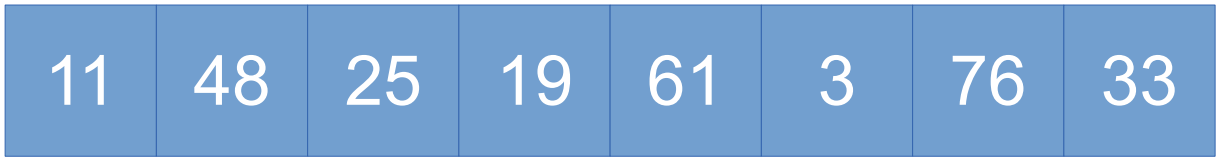
入力: X (長さ m), Y (長さ n)

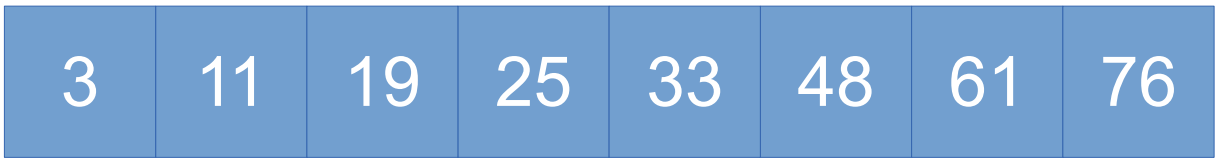
1. 長さ m+n の配列 Z を用意する
2. i = 0, j = 0 とする
3. 以下、i < m かつ j < n である限り繰り返し
   - X[i] と Y[j] を比較
     - X[i] < Y[j] ⇒ X[i] を Z[i+j] に書き、i を1増やす
     - その他 ⇒ Y[j] を Z[i+j] に書き、j を1増やす
4. i < m ならば、X[i],...,X[m-1] を Z[n+i]以降にコピー
5. j < n ならば、Y[j],...,Y[n-1] を Z[m+j]以降にコピー

マージソートのアルゴリズム

入力列Xの長さについて場合分け

- 長さ 1 ⇒ X を出力
- 長さ 1> ⇒
  - 長さ半分に切る(X1, X2とする)
  - X1, X2 をそれぞれマージソートし Y1, Y2 を得る
  - Y1, Y2 をマージした Z を出力


11
48
25
19
61
3
76
33


3
11
19
25
33
48
61
76

マージソートの時間計算量

- マージの計算量 $O(m+n)$ (m, n は入力の長さ)
- 各段で行われるマージの合計の計算量 $O(n)$ (n は要素の数)
- 段数 = $O(\log n)$

- 全体の時間計算量 = $O(n \cdot \log n)$

マージソート: その他

- 入力を保持する以外の記憶領域を使う
  - 節約するのには工夫が必要
  - ゼロにもできるがややこしい

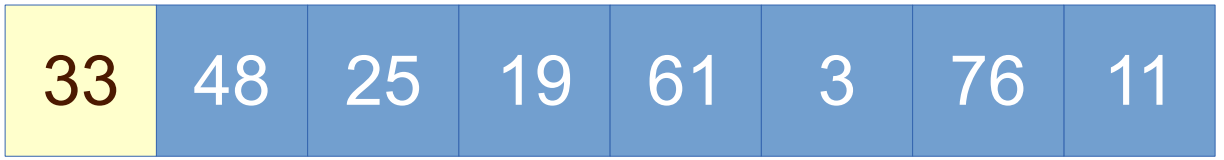
クイックソート

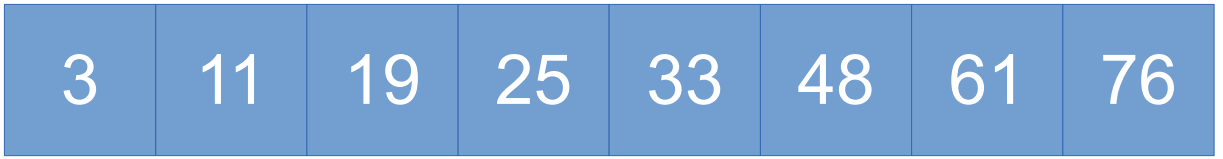
分割統治法を使った別の整列アルゴリズム

基本アイデア:

- 全体から基準値(pivot)をひとつ選ぶ
- 基準値以上の値、基準値以下の値のふたつに分割
- それぞれをクイックソート
- 得られた結果を結合

基準値




3
11
19
25
33
48
61
76

クイックソートの特徴

- 一時的な記憶領域が一切不要な分割の方法がある
  - 分割の時間計算量: $O(n)$
- ただし、基準値の選び方で分割された後のサイズ、ひいては分割段数が変わる
  - 平均時間計算量: $O(n \cdot \log n)$
  - 最悪時間計算量: $O(n^2)$
    - ただし非常に稀

おまけ
遅いソート: ボゴソート (bogosort)

- (乱数を使って)適当に並べかえる
- たまたま整列済ならOK、そうでないなら上に戻る

- 時間計算量は O(n･n!)
  - n! = n･(n-1)･(n-2)･…･2･1
- 止まるの？
  - 無限の猿定理

今日のメニュー

- 前回の復習
- ちょっとしたウンチク
- 整列アルゴリズム
- 「容易な問題」と「困難な問題」
- アルゴリズムの正しさ

問題の簡単さ・難しさ

- 易しい(tractable)…オーダーの低いアルゴリズム有
- 難しい(intractable)…オーダーの低いアルゴリズムが存在しない
- 無理…アルゴリズムがそもそも存在しない
  - 決定不能(undecidable)問題
    - チューリング機械の停止性判定

- 計算量理論での易しい/難しいの境界: 多項式オーダー
  - $O(n^2)$ も $O(n^{100})$ もやさしい!

難易度に応じたクラス分け

クラスの名前	特徴	例
決定不能 (undecidable)	アルゴリズムが存在しない	チューリング機械の停止性判定問題
EXPTIME	指数時間アルゴリズム (例: $O(2^n)$)が存在	
NP	非決定的多項式時間アルゴリズムが存在	巡回セールスマン問題、ナップザック問題
P	多項式時間アルゴリズムが存在	探索、整列

クラスNP

- 非決定的多項式時間アルゴリズムがある問題(たち)
- NP = nondeterministic polynomial time (非決定的多項式時間)
  - 非決定性…アルゴリズム中に分岐があって適当に選ばなければいけない
  - うまく分岐を選べれば易しい
    - 「てんのかみさまのいうとおり」?
  - 答えがわかれば検算は簡単
- NP 完全問題: NP の中でも難しい問題(定義は省略)

NP完全問題の例

- ナップザック問題

n 個の品物があって、それぞれ重さと価値が決まっている。重さの総和が予め定まった制限を越えない中で価値を最大にする組み合わせ(もしくは価値 v 以上の組み合わせの有無)を求める問題

- 巡回セールスマン(traveling salesman)問題

n 個の点と点間には重み(距離、移動コストのようなもの)が決まっている。全点を通って、重みが最小になる経路(もしくは重み k 以下の経路の有無)を求める問題

巡回セールスマン問題の
非決定的アルゴリズム

1. 重みの合計 k = 0 としておき、最初の点を選ぶ
2. 次の点を選ぶ
3. 重みの合計を更新する
4. 2.、3. を繰り返す
5. 全点まわったら、重みの合計をチェック

P=NP問題

- 多項式時間アルゴリズムが存在しないNP問題はまだ知られていない
  - 存在しない
    ⇒ P ≠ NP (NP問題には難しいものがある)
  - 全ての NP 問題が多項式時間で解ける
    ⇒ P = NP
    - NP完全問題のひとつがtractableと示せればよい
- ほとんどの研究者は P ≠ NP と信じているが証明はされていない
- 証明(反証でもよい)できたら100万ドル! (本当)

今日のメニュー

- 前回の復習
- ちょっとしたウンチク
- 整列アルゴリズム
- 「容易な問題」と「困難な問題」
- アルゴリズムの正しさ

アルゴリズムの正しさ

- アルゴリズムが任意の入力に対し停止するか？
- アルゴリズムがその目的を任意の入力に対して果たすか？

  - マージソート･クイックソートは本当に整列するのか？
  - 互除法はいつでも止まるのか？本当に求まったのは最大公約数なのか？

互除法の正しさの証明(1/2)

- 以下のふたつは同値
  - 相異なる自然数 m, n の GCD が g である
  - 互いに素な自然数 a, b が存在し
    m = ga かつ n = gb

互除法の正しさの証明(2/2)

- m ÷ n の余りは a ÷ b の余り c の g倍
- しかも、b と c は互いに素、つまり GCD は g

  ⇒ 入力の GCD が再帰を越えて「保存」される
- プラス、入力の和は再帰の度に減っていく

- より厳密な証明には数学的帰納法を使う

数学的帰納法

- 「任意の自然数 n について P(n)」を示したければ、以下のふたつを示せばよい
  - P(0)
  - P(k) ならば P(k+1) (任意の k について)
    - P(k) を帰納法の仮定と呼ぶ
    - ひとつ小さい数については今示そうとしている P が成立することを使ってよい

「任意の n に対し 1からnまでの和 = n(n-1)/2」の証明

- P(0): 1から0までの和 = 0(0-1)/2

- P(k) を仮定して P(k+1) を示す
  - 1 から k+1 までの和
    = 1 から k までの和 + (k+1)
    (P(k) より)
    = k(k-1)/2 + (k+1)
    = (k+1)k/2

数学的帰納法のバリエーション (累積帰納法)

- 「任意の自然数 n について P(n)」を示したければ、以下を示せばよい
  - P(0) かつ P(1) かつ ... かつP(k) ならば P(k+1) (任意の k について)
    - k 以下については P が成立しているとしてよい

累積帰納法を使った互除法の正しさの証明

「任意の n, m, k について n = m + k ならば、m, k を入力とした互除法の出力は m, k の GCD」

(証明) 累積帰納法による。m ≧ k の場合を示す。

m÷k の余り j について場合わけ:

(1) j=0の場合、出力 k は確かに m, k のGCD

(2) j>0の場合、帰納法の仮定 P(k+j) より、k, j を入力とした互除法の出力は k, j の GCD。
先般の議論より k, j のGCD = m, k のGCD。
m < k の場合も同様。(証明了)

今日のポイント

- 整列アルゴリズムいろいろ
  - 素朴にやると $O(n^2)$
  - 分割統治法で $O(n \cdot \log n)$
- Tractable/intractable な問題
- P=NP問題
- アルゴリズムの正しさの証明
  - 数学的帰納法