アルゴリズム論

2017年4月17日

樋口文人

目次

講義

- データ構造
  - データの表現
    - 負数の表現法
    - 演算との関係
      - 大小比較
      - 加法
  - 配列
    - 番号（添字）による指定
    - 名前による指定

作業

- フローチャート作成
  - 配列の中からデータを探す
    - 乱数データ
    - ランダムな順番の連続番号
  - 配列の先頭から探す
  - おまけ：でたらめに探す
- 宿題
  - 上記のプログラムの作成

負数表現

整数の表現

2進数表現	自然数	符号と絶対値	2の補数	バイアス
111	7	−3	−1	3
110	6	−2	−2	2
101	5	−1	−3	1
100	4	−0	−4	0
011	3	+3	3	−1
010	2	+2	2	−2
001	1	+1	1	−3
000	0	+0	0	−4

自然数

2進数表現	自然数
111	7
110	6
101	5
100	4
011	3
010	2
001	1
000	0

自然な数値比較

1. 最上位の桁から比較開始
2. 対象となる桁の数値が同じなら，１つ下位の桁に移る
3. 数値が異なる場合：
   その桁の数値が０の方が小さい
4. １^{0}の桁まで数値が同じ場合，
   ２つの数値は等しい

符号と絶対値

2進数表現	符号と絶対値
111	−3
110	−2
101	−1
100	−0
011	＋3
010	＋2
001	＋1
000	＋0

1. 最上位桁の符号が異なる場合，負号（1）の付いている方が小さい
2. 符号が同じ場合，絶対値部分を比較
    1. 符号が正（0）のとき自然数の大小関係と同じ
    2. 符号が負（1）のとき自然数と同様に比較，数値が異なるとき1である方を小さいとする

バイアス表現

2進数表現	バイアス
111	3
110	2
101	1
100	0
011	−1
010	−2
001	−3
000	−4

- バイアス表現されたときの数値は自然数と解釈した数値より常に一定数（ここでは4）だけ少ない
- 大小比較は自然数と同手順でよい

2の補数

2進数表現	2の補数
111	−1
110	−2
101	−3
100	−4
011	3
010	2
001	1
000	0

最上位桁は符号と同じ

1. 最上位桁が異なる場合は，1である方が小さい
2. 最上位桁が同じ場合，残りの桁について自然数と同様に比較

別解

1. 比較する2つの数値の差を計算する
2. 差の最上位桁が1のとき，被減数の方が小さい

補数の考え方（10進数で）

- 365と-365は加えると0
- 次のような xyz を考える
  ３ ＋ x ＝ ９
  ６ ＋ y ＝ ９
  ５ ＋ z ＝ ９
- x = 6, y = 3, z = 4
  634
- 1 を加える
  635
- 365 + 635 = 1000
  3桁の電卓で計算すれば…
  答えは 0

- 2進数で考えるには…
  - 例えば 101
- 各桁に最大の数字が入っていて999に相当するのは…
  - 111
    - 1 + x = 1
    - 0 + y = 1
    - 1 + z = 1
  - x = 0, y = 1, z = 0
    - 010
  - 010 + 1
    - 011
  - 101 + 011 = 1000

応用問題

浮動小数点数

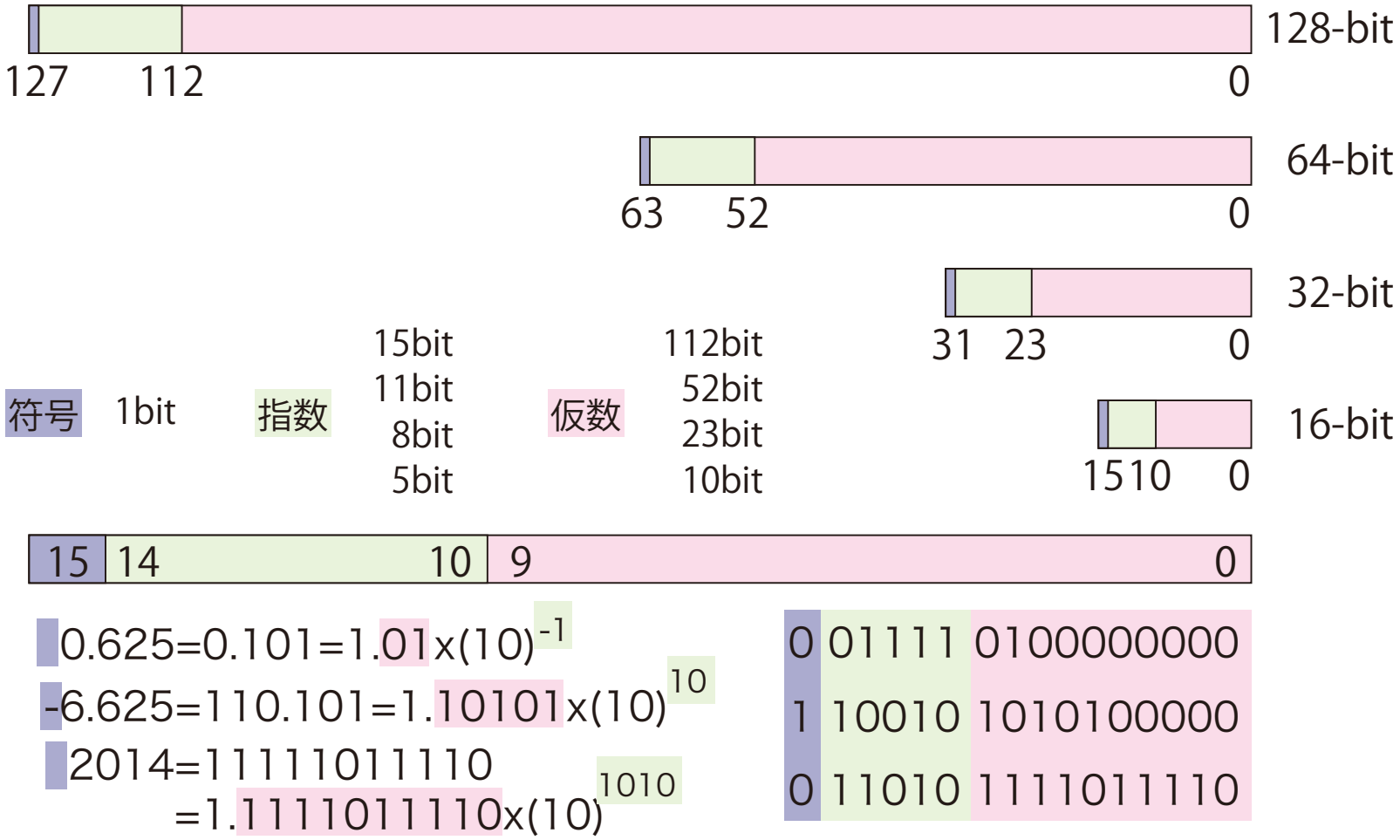
小数点のある数値の表現

仮数（Mantissa/Significand） x 基底（Base）の指数（Exponent）乗

$$\pm M \times B^{E}$$

$$0.625 = 6.25 \times 10^{-1} = 0.101_{(2)} = +1.01 \times 2^{-1}$$

IEEE 754



IEEE 754: Standard for Binary Floating-Point Arithmetic

http://grouper.ieee.org/groups/754/

浮動小数点数

- 表現している内容
  - $\pm M \times B^E$
- 表現の方法
  - ±eeeeemmmmmmmmmm (16ビットの場合)
  - 指数部：eeeeeはバイアス表現
    - 仮数が1.mm…mm の形式になるように大きさを決める
      - mm…mmの部分は固定少数点数
    - 仮数は最初の桁が必ず1なのでこれを省略
    - 小数点以下のビット並びを用いる
  - $\pm 1.mmmmmmmmmm \times 2^{(eeeee-10000)}$

$\pm M \times B^E$ の大小比較

- IEEE 754 は複数の部分からなる
  - 浮動小数点数
    - 符号＋絶対値（$M \times B^E$)
  - 絶対値
    - 仮数（M）x 基底（B=2）の指数（E）乗
- どのような手順で2つの浮動小数点数の大小を比較したら良いか？

データの配置

メモリ

メモリ空間

- アドレス（自然数）でバイト単位の指定
  - アドレス空間
  - メモリ上の情報に名前をつける：変数
    - 変数を指定する
      → メモリ上のアドレスが決まる
      → その内容が決まる
  - 情報の種類によって保存に必要なバイト数が異なる

変数

- 名前で区別
  - アルファベットで始まり，アルファベットと数字の並び
  - 区別できる変数の個数に原則制約はない
- アドレスで指定
  - 自然数で指定
  - 自然数に上限はない

指定方法の違いによる差は？

- 「名前で区別」と「アドレスで指定」
  - 同一な点
    - 指定可能な情報の個数に制約は無い
    - 順序関係がある
      - 名前をアルファベット順に並べることが可能
  - 差異のある点
    - 「次」を指定できるかどうか
      - xという変数の「次」の変数は決められない：y？　x1？　X？
    - 任意の自然数にはその次の（1だけ大きい）数が1つだけ存在
      - ある自然数とその次の数の間に他の自然数は無い

番号(自然数)による指定

- 大量のデータの集まりから個々のデータを取り出して処理する
  - 名前による個々のデータの指定
  - 番号による個々のデータの指定
- データが全て同一のデータ型であれば番号からアドレスを計算することが容易

構造を持つデータの配置

配列

配列

- 一定のデータ型を持つデータの集まり
  - 添字（と呼ばれる数値）により1つのデータを指定
    - 添字は0または1から始まる
    - 配列名[添字]という形式で指定
  - 添字の上限を予め固定する場合とそうでない場合がある

配列のサイズ

- 保持できるデータ数の上限 (n)
  - 固定されるプログラミング言語もある
  - 必要に応じて増えるプログラミング言語もある
  - 添字の上限決める
    - これ (n-1 または n) を越えるとエラーになる
    - 計算で使うデータ数mがこれより小さい (m<n) ときはプログラマが管理する必要あり

添字の有効な範囲

- 添字の有効な範囲とは
  - n 個のデータがあるとき添字の取りうる範囲は
    - 0 から n-1
    - 1 から n
- 添字が有効な範囲を逸脱すると…
  - どのような問題が起こるか

添字の利点

- データの位置情報を指定できる
  - 変数でも変数名で指定可
- 位置情報を計算によって得ることができる
  - １つの変数名から別の変数名を計算することは不可
- 連続したデータにアクセスできる
  - 変数名の連続性やある変数名の存在は曖昧
    - array[i] と array[i+1] の間にデータが存在しないことは確実
    - 変数 a と変数 b の間に他の変数はある？ない？

名前による個々のデータの指定

- 連想配列（辞書）
  - キーと値
  - キーを指定して値を得る
    - ある値を持つキーを知るのは手間が掛かる
- 配列との使い分け
  - 連想配列：処理のために1個（または少数）のデータにアクセスすれば良いとき
  - 配列：処理のために（ほとんど）全てのデータにアクセスする必要があるとき

参考

- 浅野哲夫著.
- アルゴリズム・サイエンス:入口からの超入門
  - (アルゴリズム・サイエンス シリーズ 1—超入門編)
- 共立出版, (2006)
  - **ISBN-10:** 4320121678
  - **ISBN-13:** 978-4320121676
- 6章　配列の威力
- pp. 77 – 95.

作業２

- 「１からNまでの和」を計算していた部分を「配列に格納したN個のデータの和」に修正したプログラムのフローチャートを描け
  - 配列要素に初期値を乱数で設定する
  - 和の計算はQ回繰り返す
  - Q回の繰り返し処理に掛かる時間を計測する

まとめ

- 負の数，浮動小数点数などの設計は良く出来ている
  - 比較操作（というアルゴリズム）との整合
- 配列は良く出来ている
  - データへのアクセスの利便性
  - メモリの使用効率
- プログラム言語内のデータ型とその操作
  - 掛かる時間は一定ではない
  - 同じ結果になっても効率が違う

連絡先

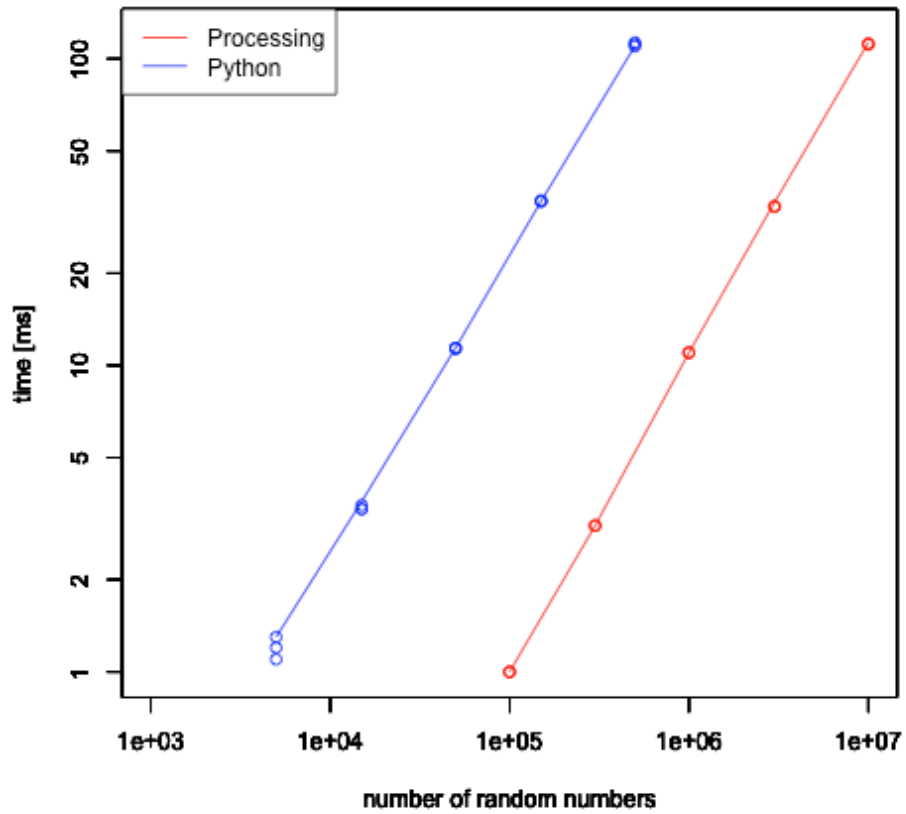
樋口文人

wenren@meiji.ac.jp

前回の宿題について

- プログラム
  - 乱数の使用
  - 二重の繰り返し
  - 繰り返し演算
  - 測定可能な問題規模の把握
  - 時刻を調べる関数の利用
    - millis()
    - System.nanoTime()

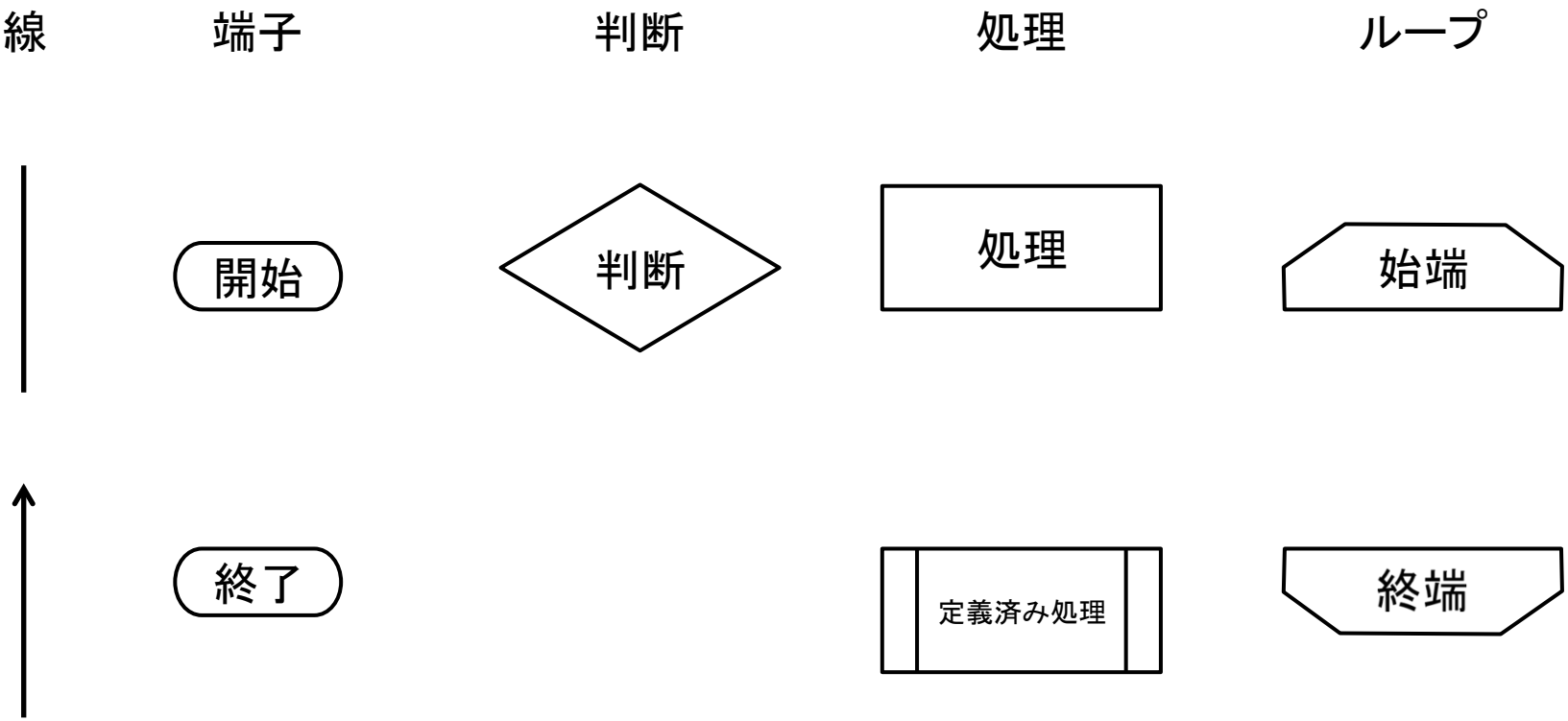
加算する乱数の個数と処理時間



ex01: 問題の規模の影響

O(N)

フローチャートの記号



やってみよう

- フローチャートを描いてください
  - 0からN-1までの数からランダムに選んだ値を以下の配列から探す手順
  - 0からN-1までの整数値をとるN個の乱数を値とする配列
  - 探索は添字0の要素から順番に行うものとする
  - 探索開始から探索の終了までの時間を計測する
  - 探索するランダムな値を新たに選んで、探索をQ回繰り返し、平均の処理時間を求める

さらに…

- 0からN-1までの連番がランダムに格納された配列を使って同様のプログラムを作る
- ２種類の配列について結果を比較する
- 探索開始から何回目の比較で見つかると期待できるか？
- さらに、さらに
  - 探索を配列の先頭から順に行うのでは無く、比較対象をランダムに選んだ場合にはどれくらい時間が掛かるか

ヒント

- 最初の配列
  - float 型のrandom(N) は[O, N) の範囲の乱数を生成する
    - floor(x) 関数は、x を内輪で最も近い整数にする
- ２番目の配列
  - 配列をdataとすると、まず data[i] = i を全ての　i について実行する
  - 次に全ての i について data[i] の値をランダムに選んだ要素 data[j] の値と交換する

宿題：ex02

- 以下の２種類のデータ配列からデータを見つけるまでの平均時間を調べて比較してください。
  1. float 型のランダムな値を持つ配列
  2. 0からN-1までの連続した整数値をランダムに並べ替えた配列
- それぞれのデータ型のランダムに選んだ値を配列の先頭から探し、その値を持つ配列要素を見つけることをQ回繰り返す時間を測定し要素を見つけるまでの平均時間を計算するプログラムを作ること
- 両者に違いはあるか？

やる気のある人に

- 配列の先頭から順に比較する対象要素を選ぶのでは無く、毎回ランダムに選ぶとするとどれくらいの時間が掛かるようになるか？
- それぞれの場合について、探索時間の期待値の振る舞いをO記法で表現できるか？

コーディング

- コメントとして:
  - プログラムの冒頭にMS, NDの別，クラス，番号，氏名を記述
  - 試した結果，考察，感想など
    - 量的にはA4レポート用紙 ½ ページ以内
    - 個人での努力が読み取れるように！
  - （人的資源を含む）参考にした資料等があれば出典を書いておく
- プログラム本体は上記コメントの後
  - プログラムは実行可能なこと

提出についての注意

- Processing のプログラム名
  - デフォルトでは sketch_yymmdda などだが…
  - higuchi_fumito_ex02 のように氏名と宿題番号に変えること
  - higuchi_fumito_c5_ex02 (同姓同名はクラスを付加）
- Processingのプログラムはフォルダごと提出
  - 必要ならzipファイルとしてまとめてください
- Oh-o!Meijiから提出（次回の授業開始までに）

python によるサンプルコード

```
import time
import random
import array
import math

Q = 1000
N = 100000
sum = 0
t_sum = 0

a = array.array('i', range(N))
for i in a:
    j = math.floor(N*random.random())
    tmp = i
    i = a[j]
    a[j]=tmp

t_start = time.clock()

for i in range(Q):
    x = math.floor(N*random.random())
    for j in range(N):
        if a[j] == x:
            break

t_sum = time.clock() - t_start

print(1000*t_sum/Q, " ms for search among", N, "data")
```

お願い

- 中村健勝
- 鈴木啓
- 永瀬緊尚
- 川島俊祐
- 古谷琢海
- 松田快
- 駒米凌弥
- 鈴木貴晴
- 田村成輝
- 野村謙介