バブルソート (その 1)

山本昌志*

2005 年 10 月 4 日

1　本日の学習内容

本日は，以下の学習を行う．

- アルゴリズムとデータ構造とは何か
- バブルソート

2　アルゴリズムとデータ構造

これからの授業では，アルゴリズムとデータ構造について学ぶ．アルゴリズムやデータ構造の実際を学ぶ前に，これらの言葉について簡単に説明しておく．

2.1　アルゴリズムとはなにか

1 年生の時に,「アルゴリズムとは、問題を解く手順である」と学習したはずである．参考文献 [1] には以下のように書かれており，より正確であろう．

> アルゴリズム (algorithm，**算法**ということもある) は，問題を解くための手順を定めたものである．この手順は，どのような操作をどのような順序で行うかを，曖昧な点の残らないようにきちんと定めたものでなければならない．
>
> 手順を明確に定めてあれば，計算機をその手順通り動かして問題を解かせることができる．アルゴリズムという概念自身は計算機と無関係に成立するが，ふつうは計算機に問題を解かせるための手順を示す．

これから，諸君は計算機で問題を解く手順を学習するのである．問題の解き方はいろいろあるけれども，効率の良いアルゴリズムを学習することになる．そして，初めてプログラムを作成する基礎が出来上がる．

効率とは何かという問題が生じるが，それは大体

- 計算の早い方が良いアルゴリズムである．

*独立行政法人　秋田工業高等専門学校　電気情報工学科

- 使用するメモリーがすくない方が良いアルゴリズムである．

と思えばよい．諸君は，計算速度を上げるためには，クロックスピードの高いコンピューターを使うことを思い浮かべるかもしれないが，それによるアップは大したことはない．クロックスピードを倍にしても，2倍程度の処理の向上しか見込めない．アルゴリズムの良し悪しにより，10 倍や 100 倍の処理速度差が瞬く間に生じることがある．処理のスピードアップには，アルゴリズムが最も重要である．

良いアルゴリズムは，経験を通して身につけるものであるが，基本的なものによく研究されている．そこで，本講義では非常に基本的なアルゴリズムを学習して，将来の応用力を身につけてもらいたい．基礎がわかっていないと，応用は絶対に不可能である．

2.2 データ構造とはなにか

データ構造 (data structure) とは，データのメモリー上の表現方法のことを言う．ただ，アセンブラ言語を除いて，直接メモリーのアドレスを指定して，データを操作するようなことはない．変数や配列，あるいは構造体というものを通して，データを操作することになる．

早いアルゴリズムを実現しようとすると，データの表現方法が重要になる．たとえば，100 個の相異なる整数があって，ある値と一致するものを探すとする．このデータをふつうの変数に入れたならば，プログラムは大変だし，それの探索 (サーチ) の方法も難しいだろう．配列に入れたならば，ある程度サーチは簡単になる．しかし，データを小さい順に整頓すればサーチはより簡単になるだろう．データが 100 個程度ならば，整列の御利益はそんなに大きくないが，100 万個くらいになると，かなりのスピードの差が生じる．値を小さい順に並べるというデータ構造にすると，格段にサーチのスピードアップが図れるのである．

このように，データを並び替えるだけで，サーチのスピードアップが図れるのであるが，並び替えの時間がかかることを忘れてはならない．ただの 1 回しか，サーチを行わないのであれば，並び替えの時間だけ無駄に終わることになる．その場合は，並び替えを行わないで，配列の端から探索する方が良いだろう．このように問題に応じて，なにが良いかを考えなくてはならない．

探索の回数が多くなると，並び替えを行ったデータ構造の方が有利であることは，明らかであろう．アルファベット順に並んでいない辞書は，使えないであろう．ただし，1 回しか単語を探さないのであれば，単語をアルファベット順に並べる方が大変である．

データの並び替えにも手間がかかるのは言うまでもない．これよりも，さらに重要なことは，整列されたデータ構造の場合，データの追加にも手間がかかるのである．これらのバランスを考えて，データ構造を構築しなくてはならない．この手間のことをコスト (cost: 費用) と表現することもある．データがバラバラに並んだ配列と，小さい順 (昇順) に並んだデータ構造を比較すると，それぞれのコストは，表 1 のようになる．

表 1: バラバラのデータと昇順に並んだデータのコストの比較

	コスト		
データ構造	並び替え	データの追加	探索
バラバラのデータ	ゼロ	きわめて低い	大
昇順のデータ	大	大	小

問題解決のためのトータルのコストが最小の場合，もっとも良いのアルゴリズムであるしデータ構造でもある．そして，最後にはコストの評価が問題となる．これは，大体計算回数と同じようなものと考えて良い．ただ，正確にこれを評価するには非常に難しい．実際には，いろいろなデータでプログラムをテストして，その実行時間を計るのがもっとも正確な方法と思われる．いつも，プログラムを作ってテストするわけにはいかないので，大体のコストの計算方法も，ここでは学習の範囲である．ただ，これについては，諸君の数学の知識では，まだ理解できないことも多いので，あまり難しいことは言わないように心がける．

3 バブルソート

3.1 手順とプログラム

ここは教科書を使って説明する．
このアルゴリズムの特徴は，

- もっとも値の大きいデータは，外側の 1 回のループで右端に移動する．
- 小さい値で，左に移動すべきデータは，一つずつしか移動できない．この一つずつしか移動できないので，整列に多くの計算回数が必要となる．
- 左に移動するデータのうち，初期位置と整列終了位置との差の最大なものにより，計算回数は決まる．

3.2 計算量

教科書のアルゴリズム (プログラム) の計算量を計算する．このプログラムの計算量は，内側のループの回数, すなわち，大小の比較回数

```
if(sort[i]>sort[i+1])
```
(1)

の実行回数で評価できるであろう．なにしろ，プログラム中もっとも実行回数の多い命令は，この文であるからである．

この文の最小実行回数は，最初から整列できている場合である．このときの，実行回数は，$N-1$ 回であるのは明らかである．最悪の場合は，最小の値が右端にある場合である．この場合の実行回数は，$(N-1)^2$ となる．なぜならば，

- この最小の値は，1 回外側のループ (do 文のところ) を実行する毎に，一つ左に移動する．
- 所定の位置に移動するために，$N-1$ 回，外側のループを実行する必要がある．
- 外側のループを 1 回実行する毎に，内側のループは $N-1$ 回，実行される．

となるからである．

最初のデータの並び方により，$N-1$ 回から $(N-1)^2$ まで開きがある．初期位置と整列終了位置との差の最大の期待値を計算することになるが，これが難しい．まだ，ちゃんとした計算が出来ていないので，正

確なことは言えないが，大体，それは $(N-2)/2$ 程度であることは分かるだろう．最小な値の初期位置と整列終了位置との差の期待値である．この程度とすると，先の比較の実行回数は，

$$\frac{(N-1)(N-2)}{2} = \frac{N^2}{2} - \frac{3}{2}N + 1 \tag{2}$$

となる．データ数 N が大きい場合，N^2 に比例して計算量が増加する．

この N^2 に比例して計算量が必要なばあい，$O(N^2)$ と書く．この表記法を O **記法**(O-notation) といい，計算オーダーを示す．数学で使うランダウの記号に似ている．

かなりいい加減な説明で，バブルソートの計算回数が $O(N^2)$ と示した．もっと正確に，説明したかったが，計算方法を思いつかなかった．ネットでも正確な記述を見つけることが出来なかった．正確な計算方法が分かれば，もう一度，説明する．

3.3 まとめ

ここで示したように，バブルソートは $O(n^2)$ の計算オーダーなので，次週で述べるクイックソートに比べて，効率の悪いアルゴリズムである．したがって，実際問題で，このソートは使ってはならないとされている．ただ，アルゴリズムが単純なので，最初の学習にはちょうど良いので取り上げられるが，ただ，使い道はそれだけである．

4 課題

4.1 課題内容

リスト 1 の続きを書いて，配列 `a[N]` に格納されている整数を昇順に並び替えるプログラムを作成すること．ただし，プログラムは，以下の要領で作成すること．

- バブルソートの部分は，独立な関数とすること．
- その関数には引数を使って，データを渡すこと．グローバル変数を使ってはならない．教科書ではグローバル変数を使っているので，そのまねをしないこと．

リスト 1: バブルソートのプログラム作成

```
1  #include <stdio.h>
2  #include <stdlib.h>     /* 乱数発生のため       */
3  #include <time.h>       /* 時刻の関数を使うため */
4  #define N 1024
5
6  int main(void){
7    int a[N], i, j, ndata, test;
8
9    srand((unsigned int)time(NULL));     /* 起動毎に異なる乱数を発生させるため */
10
11   for(i=0; i<N; i++){
12     a[i]=rand();                       /* 配列a[i]に乱数の整数を設定 */
13   }
14
```

```
15
16
17    /* これ以降にバブルソートの関数呼び出しと昇順に並んだ出力のプログラムを書く */
18
19
20    return 0;
21  }
22
23    /* ここに, バブルソートの関数を書く */
```

4.2 レポート提出要領

提出方法は、次の通りとする。

期限	10 月 17 日 (月) AM 8:50
用紙	A4
提出場所	山本研究室の入口のポスト
表紙	表紙を 1 枚つけて、以下の項目を分かりやすく記述すること。 授業科目名「情報工学」 課題名「課題　バブルソート」 2E　学籍番号　氏名 提出日
内容	ソースプログラム (プリントアウトでも、手書きでも OK とする) 作成したバブルソートの関数の引数の説明

参考文献

[1] 石畑清. アルゴリズムとデータ構造, 岩波講座 ソフトウェア科学, 3 巻. 岩波書店, 2004.