ソート (その 2)

山本昌志*

2005 年 10 月 18 日

1　本日の学習内容

本日は，以下の学習を行う．

- クイックソート
- マージンソート

2　クイックソート

2.1　手順

ここは教科書を使って説明する．

2.2　プログラムテクニック

教科書に書かれているプログラムのテクニックを説明する．main 関数から見ていって，諸君にわかりにくいものを説明する．

2.2.1　乱数

乱数とは、バラバラな数列のことを言う。とくに、自然数がめちゃくちゃに現れるようなものを自然乱数という。0 以上無限大までの全ての自然数を用いた自然乱数が考えられるが、実際上は最大の自然数を決め、その範囲で考えることが多い。0〜RAND_MAX の範囲で乱数を発生させることができる．RAND_MAX は stdlib.h に書かれており，私が使用しているシステムの C 言語の場合、

```
/* The largest number rand will return (same as INT_MAX).  */
#define RAND_MAX 2147483647
```

*独立行政法人　秋田工業高等専門学校　電気情報工学科

となっている．したがって，0〜2147483647 の範囲[1]の乱数を発生させることができるのである。

C 言語のプログラムでは、rand() 関数を使って、乱数を発生させる。教科書のプログラムでは、rand() 関数が呼び出される度に、それが乱数を返し、配列 sort[i] に格納される。

コンピューターは正確に言われたとおり (プログラムのとおり) に計算を行うことは、諸君もよく知っているはずである。そのため、コンピューターはめちゃくちゃな順序で数が並んでいる乱数を発生させることは苦手である。先ほどの rand() 関数は、ある初期値[2]を使って、計算により乱数を決めている。同じ初期値を使って、rand() 関数を呼び出すと、同じ数列が発生するこのになる。これでは、乱数とは言い難いので、初期値を毎回変更するのがよい。

初期値も値が毎回異なる整数を決める必要があるが、現在の暦時刻を返す time() 関数を用いるのが一般的である。初期値の設定は、srand() 関数に引数 (符号無し整数) を渡すことにより可能である。次のようにすれば、毎回異なる初期値を決めることができる。

ただし、1 秒以内であれば、time は同じ値となり、同じ初期値となり、同じ乱数となることに注意が必要である。(unsigned int) は、キャストと呼ばれる強制型変換で、引き続く値の型を変換している。time() 関数の引数は暦時刻で、暦時刻がポインターで格納される。暦時刻を格納する必要がないときには、NULL と空ポインターを指定する。

以上から、乱数を発生させるためには、rand() と srand()、time() 関数が必要であることが分かった。これらの関数を使うためには、関数の宣言が書かれているヘッダーファイルが必要である。rand() と srand() には stdlib.h が、time() には time.h が必要となる。

2.2.2 その他

- 関数 QuickSort のプロトタイプ宣言が書かれていない．

  関数が使われる前に，関数の定義を書くとプロトタイプ宣言を書かなくても良い．コンパイラーはソースコードのはじめから，機械語に変換するので，最初に関数の内容が分かれば，関数の使われ方があらかじめ分かる．関数を使うときに，引数のチェックが可能である．一方，関数の定義よりも前に，関数が使われると，プロトタイプ宣言がないと引数のチェックができない．

- return EXIT_SUCCESS;

  この EXIT_SUCCESS は stdlib.h に定義されているマクロで，私のパソコンでは

  ```
  #define EXIT_SUCCESS 0 /* Successful exit status.  */
  ```

  となっている．したがって，これは，

  ```
  return 0;
  ```

  と書くのと同じである．

- while 文に括弧 {} がない．

[1] 0〜$2^3$1-1 の範囲である。
[2] 正確には seed(種) と言うらしい。

プログラム中に，

```
while(lower<=upper&&data[lower]<=div)
    lower++;
```

と書かれている．これは，

```
while(lower<=upper&&data[lower]<=div){
    lower++;
}
```

と書くのと同一である．C 言語では，2 つ以上の文をひとまとめにすることがでる．これは，コードブロックまたは複文と呼ばれる構造である．コードブロックを作るには，まとめたい文を {} で囲みます．{ ここに複数の文 } で囲んだコードブロックは，ひとつの論理的なまとまりであり，単一の文が書けるところならどこでも使うことができる．

2.3 ソートの様子

実際にソートした結果を示す．

```
ソート準備:
879 269 649 711 435 311 701 605 303 879 857 419 298 278 817 713 983 469 832 531

ソート開始:
879 269 649 711 435 311 701 605 303 879 857 419 298 278 817 713 531 469 832 983
832 269 649 711 435 311 701 605 303 879 857 419 298 278 817 713 531 469 879 983

832 269 649 711 435 311 701 605 303 469 857 419 298 278 817 713 531 879 879 983
832 269 649 711 435 311 701 605 303 469 531 419 298 278 817 713 857 879 879 983
713 269 649 711 435 311 701 605 303 469 531 419 298 278 817 832 857 879 879 983

278 269 649 711 435 311 701 605 303 469 531 419 298 713 817 832 857 879 879 983

269 278 649 711 435 311 701 605 303 469 531 419 298 713 817 832 857 879 879 983

269 278 649 298 435 311 701 605 303 469 531 419 711 713 817 832 857 879 879 983
269 278 649 298 435 311 419 605 303 469 531 701 711 713 817 832 857 879 879 983
269 278 531 298 435 311 419 605 303 469 649 701 711 713 817 832 857 879 879 983

269 278 531 298 435 311 419 469 303 605 649 701 711 713 817 832 857 879 879 983
269 278 303 298 435 311 419 469 531 605 649 701 711 713 817 832 857 879 879 983

269 278 298 303 435 311 419 469 531 605 649 701 711 713 817 832 857 879 879 983

269 278 298 303 419 311 435 469 531 605 649 701 711 713 817 832 857 879 879 983

269 278 298 303 311 419 435 469 531 605 649 701 711 713 817 832 857 879 879 983

269 278 298 303 311 419 435 469 531 605 649 701 711 713 817 832 857 879 879 983

269 278 298 303 311 419 435 469 531 605 649 701 711 713 817 832 857 879 879 983
```

ソート終了:
269 278 298 303 311 419 435 469 531 605 649 701 711 713 817 832 857 879 879 983

3 マージソート

3.1 手順

ここは教科書を使って説明する.

3.2 プログラムテクニック

- `x[k++]=buffer[i++];`

  これは，後値型のインクリメント演算子を用いており，

  ```
  x[k]=buffer[i];
  i++;
  k++;
  ```

  とおなじである. 反対に, 前値型のインクリメント演算子を用いると, `x[++k]=buffer[++i];` は,

  ```
  i++;
  k++;
  x[k]=buffer[i];
  ```

  と同一の働きとなる.

- 引数で配列名を渡すところに，`x+m` となっている

  配列名は，配列の先頭を示すポインターである. したがって，配列名 `x` に整数 `m` を加算するということは，ポインターを `m` だけ移動させたことになる. したがって，`x+m` は `x[m]` を表すポインターとなっている.

3.3 ソートの様子

実際にソートした結果を示す.

ソート準備:
879 269 649 711 435 311 701 605 303 879 857 419 298 278 817 713 983 469 832 531

ソート開始:
269 879 649 711 435 311 701 605 303 879 857 419 298 278 817 713 983 469 832 531
269 879 649 435 711 311 701 605 303 879 857 419 298 278 817 713 983 469 832 531
269 879 435 649 711 311 701 605 303 879 857 419 298 278 817 713 983 469 832 531
269 435 649 711 879 311 701 605 303 879 857 419 298 278 817 713 983 469 832 531

```
269 435 649 711 879 311 701 605 303 879 857 419 298 278 817 713 983 469 832 531
269 435 649 711 879 311 701 605 303 879 857 419 298 278 817 713 983 469 832 531
269 435 649 711 879 311 701 303 605 879 857 419 298 278 817 713 983 469 832 531
269 435 649 711 879 303 311 605 701 879 857 419 298 278 817 713 983 469 832 531
269 303 311 435 605 649 701 711 879 879 857 419 298 278 817 713 983 469 832 531
269 303 311 435 605 649 701 711 879 879 419 857 298 278 817 713 983 469 832 531
269 303 311 435 605 649 701 711 879 879 419 857 298 278 817 713 983 469 832 531
269 303 311 435 605 649 701 711 879 879 419 857 278 298 817 713 983 469 832 531
269 303 311 435 605 649 701 711 879 879 278 298 419 817 857 713 983 469 832 531
269 303 311 435 605 649 701 711 879 879 278 298 419 817 857 713 983 469 832 531
269 303 311 435 605 649 701 711 879 879 278 298 419 817 857 713 983 469 531 832
269 303 311 435 605 649 701 711 879 879 278 298 419 817 857 713 983 469 531 832
269 303 311 435 605 649 701 711 879 879 278 298 419 817 857 469 531 713 832 983
269 303 311 435 605 649 701 711 879 879 278 298 419 469 531 713 817 832 857 983
269 278 298 303 311 419 435 469 531 605 649 701 711 713 817 832 857 879 879 983

ソート終了:
269 278 298 303 311 419 435 469 531 605 649 701 711 713 817 832 857 879 879 983
```

4 課題

4.1 課題内容

4.1.1 ソート

以下の数列のソートに関する問題である．これをそれぞれにソートで並び替えを行った場合，その様子を示せ．この講義ノートの「ソートの様子」で示したように記述すれば良い．ただし，様子は手書きで書くこと．

```
752 778 608 239 956 244 535 840 629 353 784 199 945 847 637 413 686 172 462 223
```

[**問 1**] バブルソートで昇順 (小さい順) の場合．この場合は，先頭から端まで 1 回のソート毎の数列の並びを書くだけでよい．

[**問 2**] クイックソートで昇順の場合

[**問 3**] マージソートで昇順の場合

4.1.2 プログラムの変更

教科書のバブルソートのプログラム (List 1-1)，クイックソートのプログラム (List 1-3)，マージソートのプログラム (List 1-5) は，いずれの場合も昇順に並び替えるようになっている．降順 (大きい順) に並び替えするようなプログラムに変更したい．以下の問いに答えよ．

[**問 4**] バブルソートのプログラム (List 1-1) のどこをどのように変更するか?

[**問 5**] クイックソートのプログラム (List 1-3) のどこをどのように変更するか?

[**問 6**] マージソートのプログラム (List 1-5) のどこをどのように変更するか?

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 24 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　ソート (その 2)」
　2E　学籍番号　氏名
　提出日
内容　2 ページ以降に問いに対する答えを分かりやすく記述すること.

5 付録

5.1 クイックソートの計算量

QuickSort() 関数が最初に呼び出されたときの比較

```
lower<=upper&&data[lower]<=div
lower<=upper&&data[upper]>div
```

の実行回数は、$N+1$ 回である。そして、再帰呼び出しにより、α 個と β 個の場合の QuickSort() が呼び出されることになる。N 個の時のトータルの比較回数を a_N として、これらの関係は、

$$a_N = N + 1 + a_\alpha + a_\beta \tag{1}$$

となる。もちろん、α と β は、1〜$N-1$ の値を取りうる。そして、sort[] に格納されている値がランダムであれば、同じ確率で 1〜$N-1$ の値をとる。さらに、α と β の和は N になることはクイックソートのアルゴリズムから自明である。したがって、a_N の期待値は、

$$\begin{aligned} a_N &= N + 1 + \frac{1}{N-1}\left(a_1 + a_2 + a_3 + \cdots + a_{N-1} + a_{N-1} + a_{N-2} + a_{N-3} + \cdots + a_1\right) \\ &= N + 1 + \frac{2}{N-1}\sum_{k=1}^{N-1} a_k \end{aligned} \tag{2}$$

となる。この漸化式から、a_N を求めることになるが、計算は結構大変である。まず、式 (2) を

$$(N-1)a_N = (N+1)(N-1) + 2\sum_{k=1}^{N-1} a_k \tag{3}$$

と変形しておく。つぎに、この式の N を $N-1$ にした場合の式

$$(N-2)a_{N-1} = N(N-2) + 2\sum_{k=1}^{N-2} a_k \tag{4}$$

を利用する。式 (3) から式 (4) の辺々を差し引いて整理すると、

$$(N-1)a_N - (N-2)a_{N-1} = (N+1)(N-1) - N(N-2) + 2a_{N-1} \tag{5}$$

となる。さらに、整理を進めると

$$(N-1)a_N - Na_{N-1} = 2N - 1 \tag{6}$$

となる。両辺を、$(N-1)N$ で割ると、

$$\begin{aligned} \frac{a_N}{N} - \frac{a_{N-1}}{N-1} &= \frac{2N-1}{(N-1)N} \\ &= \frac{1}{N} + \frac{1}{N-1} \end{aligned} \tag{7}$$

となる。ここで、a_N/N を b_N と置くと、

$$b_N - b_{N-1} = \frac{1}{N} + \frac{1}{N-1} \tag{8}$$

となり、階差数列を用いて容易に計算できる。すなわち、

$$b_N = b_2 + \sum_{k=3}^{N} \left(\frac{1}{k} + \frac{1}{k-1} \right) \tag{9}$$

である。$a_2 = 3$ より、$b_2 = 3/2$ となるので、

$$\begin{aligned} b_N &= \frac{3}{2} + \sum_{k=3}^{N} \frac{1}{k} + \sum_{k=3}^{N} \frac{1}{k-1} \\ &= \frac{3}{2} + \left(\sum_{k=1}^{N} \frac{1}{k} - 1 - \frac{1}{2} \right) + \left(\sum_{k=1}^{N} \frac{1}{k} - 1 - \frac{1}{N} \right) \\ &= -\frac{3}{2} - \frac{1}{N} + 2 \sum_{k=1}^{N} \frac{1}{k} \end{aligned} \tag{10}$$

となる。ここで、N が大きい場合、$\sum_{k=1}^{N} \frac{1}{k}$ は $\log_e N + \gamma$ に近づくことが知られている[3]。γ はオイラー数と言われる定数で、その値は $0.577\cdots$ である。したがって、N が大きい場合、

$$b_N \simeq -\frac{3}{2} - \frac{1}{N} + 2(\log_e N + \gamma) \tag{11}$$

となる。右辺の $\log_e N$ は他の項に比べて大きな値を取る[4]。したがって、

$$b_N \simeq 2 \log_e N \tag{12}$$

となる。b_N の定義から、

$$a_N \simeq 2N \log_e N \tag{13}$$

となる。

a_N は `sort[]` の要素の比較回数を表す。したがって、N が大きい場合の比較回数は、$2N \log_e N$ となる。

[3]通常は、$1 + \frac{1}{2} + \frac{1}{3} + \cdots - \log_e N = \gamma$ という関係式を憶えている

[4]通常ソートが使われるのは大きなサンプルがあるときである。$N = 10000$ を考えれば、$\log_2 N$ の項が支配的であることはすぐに分かる