選択ソートよりも駄目なソートにかんして

飯　田　浩　志*

とある講義に付随するプログラミング演習にまつわる四方山話で，アルゴリズムを習ふと最初の方に出てくる整列（sorting）にかんする話題を扱ふ。基本的な整列法に選択ソートなる手法があるが，ここでは其(そ)の選択ソートよりも明らかに手間がかかるけれども正しく動作するソートを紹介する。

キーワード：整列，プログラミング，C言語

『ソフトウエア科学』は一期四単位（週二回開講）科目であり，内半分は実際にコンピュータを用ゐたプログラミング演習を行つてゐる。使用するプログラミング言語はCである。その中で，割と早い段階で整列（ソート）を学ぶ。以下は，その整列にまつわる演習に関聯した出来事である。

整列としては，まずは選択（selection）ソートを教へてゐる*1。選択ソートは$O(N^2)$で効率がさして良くはないものの，Kernighan［1, 4.3節］でも紹介されてゐるやうに，基本的な整列法である。入力された整数Nヶを昇順（小さい順）に並べ換へて出力するには，例へば次のやうにする（此処(ここ)ではN＝5）：

```
#include <stdio.h>
#define N 5

int main(void)
{
    int a[N], i, j;

    for (i=0; i<N; i++) scanf("%d", a+i);
```

**E-mail address*: hiroggiida@me.com

*1 僕は助手で演習を手伝ふだけなので，担当の教授が。

```
    for (i=0; i<=N-2; i++)
        for (j=i+1; j<=N-1; j++)
            if (a[i] > a[j]) { /* 交換 */
                int b;

                b = a[i]; a[i] = a[j]; a[j] = b;
            }
    for (i=0; i<N; i++) printf("%d ", a[i]);
    putchar('\n');
    return 0;
}
```

ひとつ，Cでは配列の添字は0から始まることに注意されたい*2。以上の処理での整列はつまり，外側iにかんするforループでi=0の処理（i=0固定の下，内側jのforループ）が終ると，a[0]に最小値が収まることになる。次に，a[0]のことはもう忘れて切り離し，i=1の処理で，二番目に小さい数値がa[1]に来る。此れをi=N−2まで行つて，昇順になつた列を得る*3。

さて以前，演習の時間に，この整列部を以下のとおり書いて持つてきた学生さんが居たやうに記憶してゐる：

```
for (i=0; i<N; i++)
    for (j=0; j<N; j++)
        if (a[i] < a[j]) {
            int b; b = a[i]; a[i] = a[j]; a[j] = b;
        }
```

*2 int a[100]；と宣言すれば，a[0]からa[99]まで100ヶのintが使える。ちょっと余談，配列の添字が0 originだと，剰余系との相性が良い。例へば，x月からyヶ月先は何月か？といふ問題を考へる。答を言つてしまふと（x+y−1）%12+1なのだが（%は余りを計算するCの演算子）もしカレンダーが0月から11月であれば（x+y）%12で済む。また，配列の添字が0から始まることで，リングバッファの管理等も容易になる。

*3 実害は無いものの，外側forの終了条件をi<=N−1（もしくはi<N）とする学生さんが居る。最後の一つは最大値が残るに決まってるし，そもi=N−1の時，内側のforはjがNからN−1までとなって，一回も回らない——つまりは，比較対象が無い。

そもそも，交換するか否かを判定する不等号の向きが逆だし，しょっぱなi=0，j=0は同じものを比較するといふことで意味がない。これらのことからも，これが動作する筈がないとずっと思つてゐたのだが，ふと思い立つて実行してみたら，見事に昇順に整列できた。以下では，なぜ此れで整列されるのか，を説明する。以降の記述で，配列は左から右に伸びてゐるとイメージされたい。

先ず，左端a[0]は最小値で確定。なぜなら，当初a[k(>0)]に最小があったなら，i=k，j=0でa[0]に移動して以降は動かないし，特に最初からa[0]が最小なら，i=0，j=1の時にa[1]に移動して，後のi=1，j=0で最大値と交換されてa[0]に戻つて（其れでi=1の処理は終了，あとは何も起こらない）以降は不動だから。

さらに，各iにかんして内側のforループ終了時，最大値は常にa[i]にあるので，最大が一番右a[N−1]に来るは自明。では，二番目に最大が，最後の交換（i=N−1，j=N−2）の時に一番右に居てるか（i=N−2の終了時，最大値はa[N−2]にある）といへば，そのとおり。何故なら，最後の交換の直前，a[N−1]に居るのは，最大値a[N−2]を除いた中で一番大きいから。よってa[N−2]，a[N−1]も正しい数値を保持してゐる。

特にN=4では，上の議論から，最終的にa[1]以外が正しい位置にあること確定につき，整列されてゐると云へる。ではN=5は？帰納法による証明になる？

ぢつは勝手なiに就て，内側のforループ終了時点でa[0]，a[1]，…，a[i]は昇順に整列してゐる。それが凡て。無駄を削ぎ落とせば，以下になる（要は，挿入ソート）——つまり，a[i]の這入るべき位置をa[0]，a[1]，…，a[i]から探して抉じ入れる（1⩽i⩽N−1）：

```
for (i=1; i<N; i++)
    for (j=0; j<i; j++)
        if (a[i] < a[j]) {
            int b; b = a[i]; a[i] = a[j]; a[j] = b;
        }
```

外側のforは i＝1 から始めて構はない。a[0]ひとつは,既に整列済と云へる。加へて，j>i の所で内側のforを回しても a[i]がより大きくなるやもと云ふだけで a[0]，a[1]，…，a[i]が整列済であることは崩れぬから，本質的な操作ではない*4。

もし a[j]＞a[i]になったら，a[i]を入れる位置が見つかったことになる。ここで，a[j]から後を一つずつずらして a[j]の位置に a[i]を入れる方が分かりやすいかも，i.e.：

```
if (a[j] > a[i]) { /* found a position where a[i] should be */
    int b, k;

    b = a[i];
    for (k=i; k>j; k--) a[k] = a[k-1];
    a[j] = b;
    break; /* do next i */
}
```

ここにbreakは効率の為であり，無くても動作する*5。

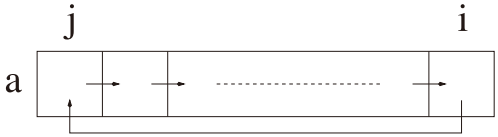

図１　a[i]を正しい位置に嵌(は)め込む

*4 精確には，i＝1 の時に a[1]が最大値になって以降は，各 i（≧２）に就て，a[i]が最大だから j>i では何も起きず。

*5 挿入ソートの実装には此の他，１から始めてＮ−１までの各 i に就て，バブルソートの如く a[i]が左隣の大きい間は交換しつつ左へと順次移動していく（無論，左端は超えず）と云ふのもある。

余談になるけれど，函数の先頭で凡ての変数を宣言する，と云ふスタイルをよく見かけるが，上のやうに if 文でローカルな変数は if 文の中で宣言すれば佳(よ)い。C では，（C++やJavaほどには自由ではないけれども）ブロックの先頭で変数を宣言できる*6。特に，函数の先頭で宣言し且(か)つ初期化した変数など，それが登場する所がかなり先なら，初期化したことは疎(おろ)か宣言したことすら覚へてはいまい。であれば，変数を使ふ直前で宣言する方が親切だし，変数が当該ブロックのみでローカルなら他には影響しないと云ふscope rule上の利点もある，と拝察するが如何。

参考文献

[1] Brian W. Kernighan, *Understanding the Digital World*. Princeton University Press, 2017.

追記：選択ソートと書きましたが単純ソートとのこと。選択ソートは i=0 で最小を覚えておき最後に左端と交換して次 i=1 で残り中の最小を，と交換は＜N回。例へば 2，3，4，5，1 だと 4(＝N－1)回。

*6 規格C99では，変数宣言がブロックの先頭といふ制限が廃止されたとか。でも其れって，どうなのかなぁ。