

第9章 C言語 応用編：ソーティング

9.1 ソーティング

コンピュータは、大量のデータを高速かつ正確に処理することを得意としますが、その中でも、データの検索・追加・削除・並び替えといった操作はデータ処理の基本となります。この章では、並び替え、すなわち**ソーティング** (sorting) に絞って、そのアルゴリズムを紹介します。

ソーティングは、与えられた n 個のデータの列 $\{a_1, a_2, \dots, a_n\}$ をある基準に従って順に並び替えることです¹。並べ替える基準には、データを $a'_1 \leq a'_2 \leq \dots \leq a'_n$ のように単調増加的に並べる**昇順ソート** (ascending sort) と、データを $a'_1 \geq a'_2 \geq \dots \geq a'_n$ のように単調減少的に並べる**降順ソート** (descending sort) があります。

有名なソーティングアルゴリズムには、

- 9.2 節 **選択ソート** (selection sort) * **最小値選択法**とも呼ばれる。
- 9.3 節 **バブルソート** (bubble sort)
- 9.4 節 **挿入ソート** (insertion sort) * 挿入ソートを拡張した**シェルソート** (shell sort) がある。
- 9.5 節 **クイックソート** (quick sort)
- 9.6 節 **ヒープソート** (heap sort)
- 9.7 節 **マージソート** (merge sort) * **併合ソート**とも呼ばれる。

¹ここで言うある基準とは、データの大小の比較によってソーティングを行なうことです。なお、本テキストでは紹介しませんが、**ビンソート** (bin sort)・**分布数え上げソート** (distribution counting sort)・**基数ソート** (radix sort) など、比較によらないソーティング方法もあります。

があります。

なお、これらアルゴリズムはその性格によって、データ列を主記憶装置 (メモリ) に置いてソーティングするのに適した**内部ソート** (internal sort; **内部整列**) と、データ列をテープやハードディスクなどの外部装置に置いてソーティングするのに適した**外部ソート** (external sort; **外部整列**) に分類されます²。上記に挙げたアルゴリズムの中では、唯一、マージソートが外部整列に属しています。

以後、各アルゴリズムの解説を行います。データ列には {4, 10, 5, 2, 1, 7, 8, 6, 3, 9} を具体例として使用し、

$$\{4, 10, 5, 2, 1, 7, 8, 6, 3, 9\} \implies \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

のように昇順にソーティングされて行く過程を観察します。ただし、C言語によるプログラムでは、添え字は0から始めるのが普通ですが理解しやすいように添え字を1から始めてあります (配列を「`int a[10+1]={0,4,10,5,2,1,7,8,6,3,9};`」と宣言してあります)。また、各ソーティングアルゴリズムについて、少し安直ですが比較回数を基準に**計算量**のオーダを導き、アルゴリズムの良し悪しについても考察します (1回の比較を1演算として換算します)。

9.2 選択ソート

選択ソート (最小値選択法) は、未整列部分のデータ列の最小値を選択し、未整列部分の先頭と最小値を交換する操作を繰り返し実行することでソーティングを行ないます。アルゴリズムは、

- Step 1** $a_i = \min(a_1, a_2, \dots, a_n)$ を探し、 a_i と a_1 を交換する。 a_1 が決定する。
 * $\min(a_1, a_2, \dots, a_n)$ を決定するために $n-1$ 回の比較が必要となる。
- Step 2** $a_j = \min(a_2, a_3, \dots, a_n)$ を探し、 a_j と a_2 を交換する。 a_2 が決定する。
 * $\min(a_2, a_3, \dots, a_n)$ を決定するために $n-2$ 回の比較が必要となる。
- $\vdots$
- Step $n-1$** $a_k = \min(a_{n-1}, a_n)$ を探し、 a_k と a_{n-1} を交換する。 a_{n-1} と a_n が決定する。
 * $\min(a_{n-1}, a_n)$ を決定するために 1 回の比較が必要となる。

となります。従って、データ列 {4, 10, 5, 2, 1, 7, 8, 6, 3, 9} を選択ソートによってソーティングすると表 9.1 のようになります。

なお、選択ソートでは、Step 1 で $n-1$ 回、Step 2 で $n-2$ 回、 $\dots$ 、Step $n-1$ で 1 回の比較が必要となるので、計算量は

$$(n-1) + (n-2) + \dots + 1 = \frac{\{1 + (n-1)\} \cdot (n-1)}{2} = \frac{n \cdot (n-1)}{2}$$

となります。更に、オーダ記法で表すと $O(n^2)$ となります。

²データ列が主記憶装置に収まらない場合は、データ列を主記憶装置に収まるように分け、分けた各データ列を内部整列でソーティングし、外部整列で各データ列を結合します。現実的には、このようにアルゴリズムを組み合わせるソーティングするのが一般的です。

START	{4, 10, 5, 2, 1, 7, 8, 6, 3, 9}	
Step 1	{4, 10, 5, 2, 1, 7, 8, 6, 3, 9}	未整列部分から最小値を探し、
	{1, 10, 5, 2, 4, 7, 8, 6, 3, 9}	最小値 1 と a_1 を交換する。
	{1, 10, 5, 2, 4, 7, 8, 6, 3, 9}	a_1 が決定する。
Step 2	{1, 10, 5, 2, 4, 7, 8, 6, 3, 9}	未整列部分から最小値を探し、
	{1, 2, 5, 10, 4, 7, 8, 6, 3, 9}	最小値 2 と a_2 を交換する。
	{1, 2, 5, 10, 4, 7, 8, 6, 3, 9}	a_2 が決定する。
Step 3	{1, 2, 3, 10, 4, 7, 8, 6, 5, 9}	以下、同様の手順を繰り返す。
Step 4	{1, 2, 3, 4, 10, 7, 8, 6, 5, 9}	
Step 5	{1, 2, 3, 4, 5, 7, 8, 6, 10, 9}	
Step 6	{1, 2, 3, 4, 5, 6, 8, 7, 10, 9}	
Step 7	{1, 2, 3, 4, 5, 6, 7, 8, 10, 9}	
Step 8	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}	
Step 9	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}	
END	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}	

表 9.1: 選択ソートによるソーティングの例

問題 1 「選択ソートによるソーティングプログラム」を作成しなさい。

9.3 バブルソート

バブルソート³は、隣り合う 2 つの項を比較しながらデータ列を左から右へ走査することでソーティングを行ないます。ただし、隣り合う 2 つの項を比較したとき 2 つのデータが並べ替えの基準を満たさなければデータを交換するという作業を行います。結果的には、データ列の右から順にソーティングが完了して行きます。アルゴリズムは、

Step 1	a_i と a_{i+1} を比較し、 $a_i > a_{i+1}$ ならば交換する ($i = 1, 2, \dots, n-1$)。 a_n が決定する。 * $n-1$ 回の比較が必要となる。
Step 2	a_i と a_{i+1} を比較し、 $a_i > a_{i+1}$ ならば交換する ($i = 1, 2, \dots, n-2$)。 a_{n-1} が決定する。 * $n-2$ 回の比較が必要となる。
$\vdots$	$\vdots$
Step $n-1$	$a_i (= a_1)$ と $a_{i+1} (= a_2)$ を比較し、 $a_1 > a_2$ ならば交換する ($i = 1$)。 a_1 と a_2 が決定する。 * 1 回の比較が必要となる。

となります。従って、データ列 {4, 10, 5, 2, 1, 7, 8, 6, 3, 9} をバブルソートによってソーティングすると表 9.2～表 9.3 のようになります。

³ソーティングによりデータが確定していく様子が、泡が下から上昇している様子に似ていることからバブルソートと呼ばれます。

START	{4, 10, 5, 2, 1, 7, 8, 6, 3, 9}	
Step 1	{4, 10, 5, 2, 1, 7, 8, 6, 3, 9}	a_1 と a_2 を比較し、
	{4, 10, 5, 2, 1, 7, 8, 6, 3, 9}	$a_2 \leq a_3$ なのでそのまま。
	{4, 10, 5, 2, 1, 7, 8, 6, 3, 9}	a_2 と a_3 を比較し、
	{4, 5, 10, 2, 1, 7, 8, 6, 3, 9}	$a_2 > a_3$ なので交換する。
	{4, 5, 2, 10, 1, 7, 8, 6, 3, 9}	•
	{4, 5, 2, 1, 10, 7, 8, 6, 3, 9}	•
	{4, 5, 2, 1, 7, 10, 8, 6, 3, 9}	•
	{4, 5, 2, 1, 7, 8, 10, 6, 3, 9}	•
	{4, 5, 2, 1, 7, 8, 6, 10, 3, 9}	•
	{4, 5, 2, 1, 7, 8, 6, 3, 10, 9}	•
	{4, 5, 2, 1, 7, 8, 6, 3, 9, 10}	(a_{10} が決定する。)
Step 2	{4, 5, 2, 1, 7, 8, 6, 3, 9, 10}	以下、Step 1 と同様の手順を繰り返す。
	{4, 2, 5, 1, 7, 8, 6, 3, 9, 10}	
	{4, 2, 1, 5, 7, 8, 6, 3, 9, 10}	
	{4, 2, 1, 5, 7, 8, 6, 3, 9, 10}	
	{4, 2, 1, 5, 7, 8, 6, 3, 9, 10}	
	{4, 2, 1, 5, 7, 6, 8, 3, 9, 10}	
	{4, 2, 1, 5, 7, 6, 3, 8, 9, 10}	
	{4, 2, 1, 5, 7, 6, 3, 8, 9, 10}	(a_9 が決定する。)
Step 3	{2, 4, 1, 5, 7, 6, 3, 8, 9, 10}	
	{2, 1, 4, 5, 7, 6, 3, 8, 9, 10}	
	{2, 1, 4, 5, 7, 6, 3, 8, 9, 10}	
	{2, 1, 4, 5, 7, 6, 3, 8, 9, 10}	
	{2, 1, 4, 5, 6, 7, 3, 8, 9, 10}	
	{2, 1, 4, 5, 6, 3, 7, 8, 9, 10}	
	{2, 1, 4, 5, 6, 3, 7, 8, 9, 10}	(a_8 が決定する。)
Step 4	{1, 2, 4, 5, 6, 3, 7, 8, 9, 10}	
	{1, 2, 4, 5, 6, 3, 7, 8, 9, 10}	
	{1, 2, 4, 5, 6, 3, 7, 8, 9, 10}	
	{1, 2, 4, 5, 6, 3, 7, 8, 9, 10}	
	{1, 2, 4, 5, 3, 6, 7, 8, 9, 10}	
	{1, 2, 4, 5, 3, 6, 7, 8, 9, 10}	(a_7 が決定する。)

表 9.2: バブルソートによるソーティングの例 (前半)

Step 5	{1, 2, 4, 5, 3, 6, 7, 8, 9, 10}
	{1, 2, 4, 5, 3, 6, 7, 8, 9, 10}
	{1, 2, 4, 5, 3, 6, 7, 8, 9, 10}
	{1, 2, 4, 3, 5, 6, 7, 8, 9, 10}
	{1, 2, 4, 3, 5, 6, 7, 8, 9, 10} (a_6 が決定する。)
Step 6	{1, 2, 4, 3, 5, 6, 7, 8, 9, 10}
	{1, 2, 4, 3, 5, 6, 7, 8, 9, 10}
	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10} (a_5 が決定する。)
↑ この Step でソーティングが終了している。	
Step 7	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10} (a_4 が決定する。)
Step 8	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10} (a_3 が決定する。)
Step 9	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10} (a_2 と a_1 が決定する。)
END	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

表 9.3: バブルソートによるソーティングの例 (後半)

なお、バブルソートでは、Step 1 で $n-1$ 回、Step 2 で $n-2$ 回、 $\dots$ 、Step $n-1$ で 1 回の比較が必要となるので、計算量は

$$(n-1) + (n-2) + \dots + 1 = \frac{\{1 + (n-1)\} \cdot (n-1)}{2} = \frac{n \cdot (n-1)}{2}$$

となります。更に、オーダ記法で表すと $O(n^2)$ となります。

ところで、表 9.3 をよく見るとソーティングは Step 6 で終了しています。すなわち、以降のステップを省略することができます (終了条件は次のステップで交換が起これなければ、その時点でソーティングを終了します)。特に、完全に整列済みのデータ列であれば Step 1 で終了条件を直ちに満たすため比較回数は $n-1$ 回となり、計算量は $n-1$ となります (オーダ記法で表すと $O(n)$ となります)。また、ほとんど整列済みのデータ列の場合も (Step i ($\ll n$) で終了条件を満たすとする)、計算量は

$$(n-1) + (n-2) + \dots + (n-i) = \frac{\{(n-1) + (n-i)\} \cdot i}{2} = \frac{(2n-i-1) \cdot i}{2}$$

となり、オーダ記法で表すと $O(n)$ となります⁴ ($\because i$ は具体的に与えられた小さな数であるから、 $O(i \cdot n) \Rightarrow O(n)$ となる)。この様に、バブルソートはデータ列に依存したアルゴリズムです。

問題 1 「バブルソートによるソーティングプログラム」を作成しなさい。

⁴ほとんど整列済みのデータ列でも、 $\{2, 3, 4, 5, 6, 7, 8, 9, 10, 1\}$ のような場合は $O(n^2)$ となります。ただし、アルゴリズムを「 a_i と a_{i+1} を比較し、 $a_i > a_{i+1}$ ならば交換する ($i = n-1, n-2, \dots, 1$)」と変更すれば $O(n)$ となります。

9.4 挿入ソート

挿入ソートは、ある位置まで整列済みのデータ列に未整列の要素を適切な位置に挿入する操作を繰り返し実行することでソーティングを行ないます。アルゴリズムは、

- Step 1** 整列済みのデータ列 $\{a_1\}$ に未整列の要素 a_2 を適切な位置に挿入する。
 * 挿入位置を決めるため最小で1回、最大で1回の比較が必要となる。
- Step 2** 整列済みのデータ列 $\{a_1, a_2\}$ に未整列の要素 a_3 を適切な位置に挿入する。
 * 挿入位置を決めるため最小で1回、最大で2回の比較が必要となる。
- $\vdots$
- Step $n-1$** 整列済みのデータ列 $\{a_1, a_2, \dots, a_{n-1}\}$ に未整列の要素 a_n を適切な位置に挿入する。
 * 挿入位置を決めるため最小で1回、最大で $n-1$ 回の比較が必要となる。

となります。従って、データ列 $\{4, 10, 5, 2, 1, 7, 8, 6, 3, 9\}$ を挿入ソートによってソーティングすると表 9.4 のようになります。

START	$\{4, 10, 5, 2, 1, 7, 8, 6, 3, 9\}$	
Step 1	$\{\{4\}, 10, 5, 2, 1, 7, 8, 6, 3, 9\}$ $\{\{4, 10\}, 5, 2, 1, 7, 8, 6, 3, 9\}$	整列済みデータ列に未整列要素を適切な位置に挿入する。
Step 2	$\{\{4, 10\}, 5, 2, 1, 7, 8, 6, 3, 9\}$ $\{\{4, 5, 10\}, 2, 1, 7, 8, 6, 3, 9\}$	整列済みデータ列に未整列要素を適切な位置に挿入する。
Step 3	$\{\{4, 5, 10\}, 2, 1, 7, 8, 6, 3, 9\}$ $\{\{2, 4, 5, 10\}, 1, 7, 8, 6, 3, 9\}$	以下、同様の手順を繰り返す。
Step 4	$\{\{2, 4, 5, 10\}, 1, 7, 8, 6, 3, 9\}$ $\{\{1, 2, 4, 5, 10\}, 7, 8, 6, 3, 9\}$	
Step 5	$\{\{1, 2, 4, 5, 10\}, 7, 8, 6, 3, 9\}$ $\{\{1, 2, 4, 5, 7, 10\}, 8, 6, 3, 9\}$	
Step 6	$\{\{1, 2, 4, 5, 7, 10\}, 8, 6, 3, 9\}$ $\{\{1, 2, 4, 5, 7, 8, 10\}, 6, 3, 9\}$	
Step 7	$\{\{1, 2, 4, 5, 7, 8, 10\}, 6, 3, 9\}$ $\{\{1, 2, 4, 5, 6, 7, 8, 10\}, 3, 9\}$	
Step 8	$\{\{1, 2, 4, 5, 6, 7, 8, 10\}, 3, 9\}$ $\{\{1, 2, 3, 4, 5, 6, 7, 8, 10\}, 9\}$	
Step 9	$\{\{1, 2, 3, 4, 5, 6, 7, 8, 10\}, 9\}$ $\{\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}\}$	
END	$\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$	

表 9.4: 挿入ソートによるソーティングの例

なお、挿入ソートでは、挿入位置を決定するために、Step 1 で最大 1 回、Step 2 で最大 2 回、
 …、Step $n-1$ で最大 $n-1$ 回の比較が必要となるので、計算量は

$$1 + 2 + \cdots + (n-1) = \frac{\{1 + (n-1)\} \cdot (n-1)}{2} = \frac{n \cdot (n-1)}{2}$$

となります。更に、オーダ記法で表すと $O(n^2)$ となります。

挿入ソートもデータ列によって計算量変動するアルゴリズムです。最も良いデータ列は、挿入位置を決定するのにどの Step でも比較回数が 1 回で済む場合 (非常に稀なケース) で、その計算量は $n-1$ となります。一般には、挿入位置が平均的に (Step i において等確率 $\frac{i}{2}$ 回の比較で) 見つかりと仮定して、計算量

$$\frac{1}{2} + \frac{2}{2} + \cdots + \frac{n-1}{2} = \frac{(\frac{1}{2} + \frac{n-1}{2}) \cdot (n-1)}{2} = \frac{n \cdot (n-1)}{4}$$

を求めると、やはりオーダは $O(n^2)$ となります。

問題 1 「挿入ソートによるソーティングプログラム」を作成しなさい。

問題 2 1, 2, …, 10 で構成されるデータ列の内、挿入ソートを用いると計算量が最大となるデータ列と最小となるデータ列をそれぞれ求めなさい。

9.5 クイックソート

クイックソートは、データ列から適切な基準値を選び、その基準値より小さな要素で構成されたデータ列と大きな要素で構成されたデータ列に分割することを繰り返し実行することでソーティングを行ないます (分割は、分割されたデータ列の要素の個数が 1 個になるまで再帰的に行なう)。アルゴリズムは、

Step 1 $\{a_1, a_2, \dots, a_n\}$ から適切な基準値を選ぶ (ここでは a_1 を選ぶ)。基準値より小さな要素 ($a'_i < a_1, i = 1, \dots, j$) で構成されたデータ列 $\{a'_1, a'_2, \dots, a'_j\}$ と大きな要素 ($a''_i \geq a_1, i = 1, \dots, k$) で構成されたデータ列 $\{a''_1, a''_2, \dots, a''_k\}$ に分割する ($j + k = n$)。

* 基準値と n 回の比較が必要となる。ただし、以降の Step では、要素の個数が 1 個の分割されたデータ列が含まれていれば、そのデータ列では比較する必要がないため、最大で n 回となる。

Step 2 $\{a'_1, a'_2, \dots, a'_j\}$ から基準値 a'_1 を選び、Step 1 と同様に分割する。

$\{a''_1, a''_2, \dots, a''_k\}$ から基準値 a''_1 を選び、Step 1 と同様に分割する。

* 基準値と最大で n 回の比較が必要となる (基準値 a'_1 と j 回 + 基準値 a''_1 と k 回)。

⋮ ⋮

Step m 分割された各データ列の要素が全て 1 個になれば終了。

* 基準値と最大で n 回の比較が必要となる。

注意： このアルゴリズムの Step 数は、再帰的方法による深さを表します。