アルゴリズムとデータ構造

第11週目

担当　情報システム部門　徳光政弘
2025年9月24日

今日の内容

- 挿入ソート
- 選択ソート
- クイックソート

ソートの定義

学籍番号	氏名	点数
3001	石川	60
3002	川上	65
3003	中村	90
3004	深川	85
3005	野中	70

点数順に
ソート
→

学籍番号	氏名	点数
3003	中村	90
3004	深川	85
3005	野中	70
3002	川上	65
3001	石川	60

図 5.1　ソートの例

◆定義 5.1　ソート

ソートとは，入力として，全順序関係が定義されている n 個のデータ $d_0, d_1, \ldots, d_{n-1}$ が与えられたときに，そのデータを全順序関係に従って並べ替える操作である．

ソートと全順序関係

◈定義 5.2　全順序関係

順序関係とは，データの大小関係のことで，すべてのデータに対して以下の性質が成り立つ場合，関係 “$\leq$” は順序関係である．

反射則　すべての x について，$x \leq x$ が成り立つ．

推移則　すべての x, y, z について，$x \leq y$ かつ $y \leq z$ ならば，$x \leq z$ が成り立つ．

反対称則　すべての x, y について，$x \leq y$ かつ $y \leq x$ ならば，$x = y$ が成り立つ．

くわえて，すべてのデータの対に対して，順序関係 “$\leq$” が以下の性質をもつとき，その順序関係を全順序関係であるという．

比較可能性　すべての x, y について，$x \leq y$ もしくは $y \leq x$ が成り立つ．

ソートの入出力

入力：$\{17, 39, 1, 9, 5, 24, 2, 11, 23, 6\}$

出力：$\{1, 2, 5, 6, 9, 11, 17, 23, 24, 39\}$

ソートアルゴリズムに、ソート対象のデータを入力すると、データが並べ替えられる

基本的なソート（選択ソート）

① 入力データの中から最大のデータをみつける．

② みつけた最大のデータをソートの対象から除外する．

③ ①，②の操作を $n-1$ 回繰り返す．

アルゴリズム 5.1　選択ソート

入力：サイズ n の配列 D[0], D[1], ..., D[n-1]

```
for (i=n-1; i>0; i=i-1) {
  max=D[0]; max_index=0;
  for (j=1; j<=i; j=j+1) {
    if (D[j]>=max) { max=D[j]; max_index=j; }
  }
  swap(D[max_index],D[i]);
}
```

選択ソートの実行例

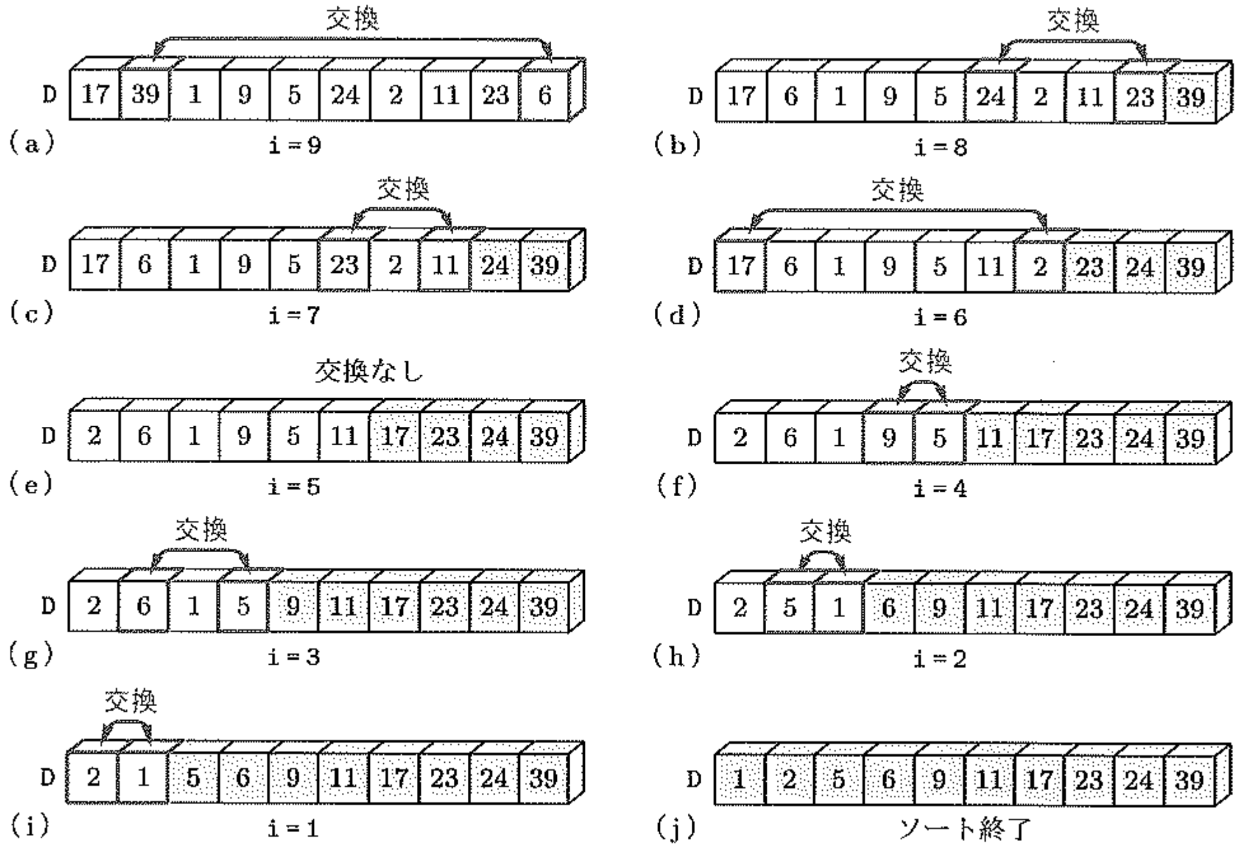


図 5.2　選択ソートの実行例

計算量

- 外側のループで比較はn-1回
- 内側のループはi回
- 数列を整理すると、計算量が求まる（公式）

$$\sum_{i=1}^{n-1} i \times O(1) = O(1) \times \frac{n(n-1)}{2}$$

$$= O(n^2)$$

挿入ソート

- トランプカードの手札を並べ替えるときに、人間がやりやすい方法

① 左手にすでに並んだ状態のカードをもつ（最初は，１枚のカードから始める）．

② 並べたい１枚のカードを右手に持ち，すでに並んでいる左手のカードの数字を右から左へ見て，カードが挿入されるべき場所を探す．

③ 右手のカードを左手の並んだカードに挿入する．

挿入ソート

アルゴリズム 5.2　挿入ソート

入力：サイズ n の配列 D[0], D[1], ..., D[n-1]

```
for (i=1; i<n; i=i+1)  {
  x=D[i]; j=i;                       //D[i]を挿入する値を表す変数xに設定
  while ((D[j-1]>x)かつ(j>0)) {      //挿入する値とD[j-1]を比較
    D[j]=D[j-1];                     //D[j-1]のほうが大きければ，値を右にずらす
    j=j-1;
  }
  D[j]=x;
}
```

挿入ソート

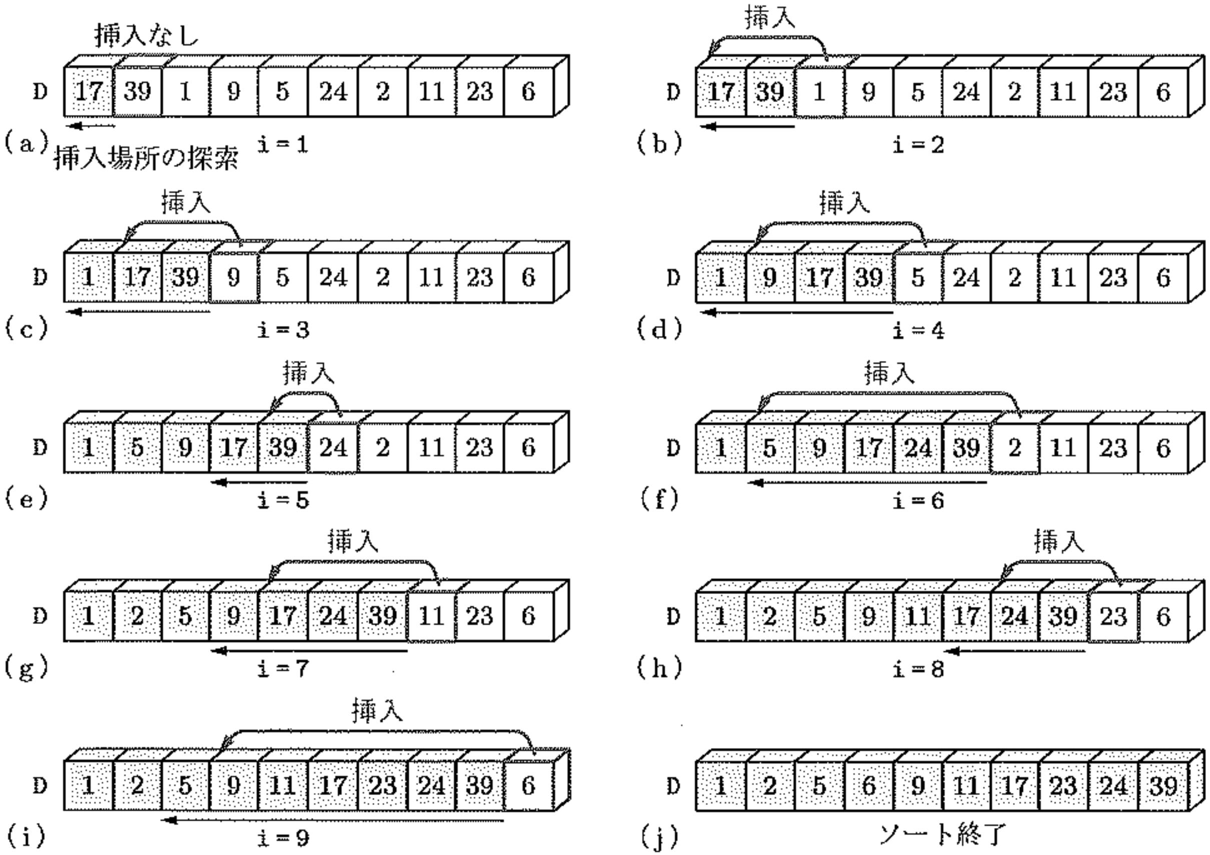


図 5.3　挿入ソートの実行例

計算量

最良計算量の場合は、挿入操作が発生しないため、O(n)となる。

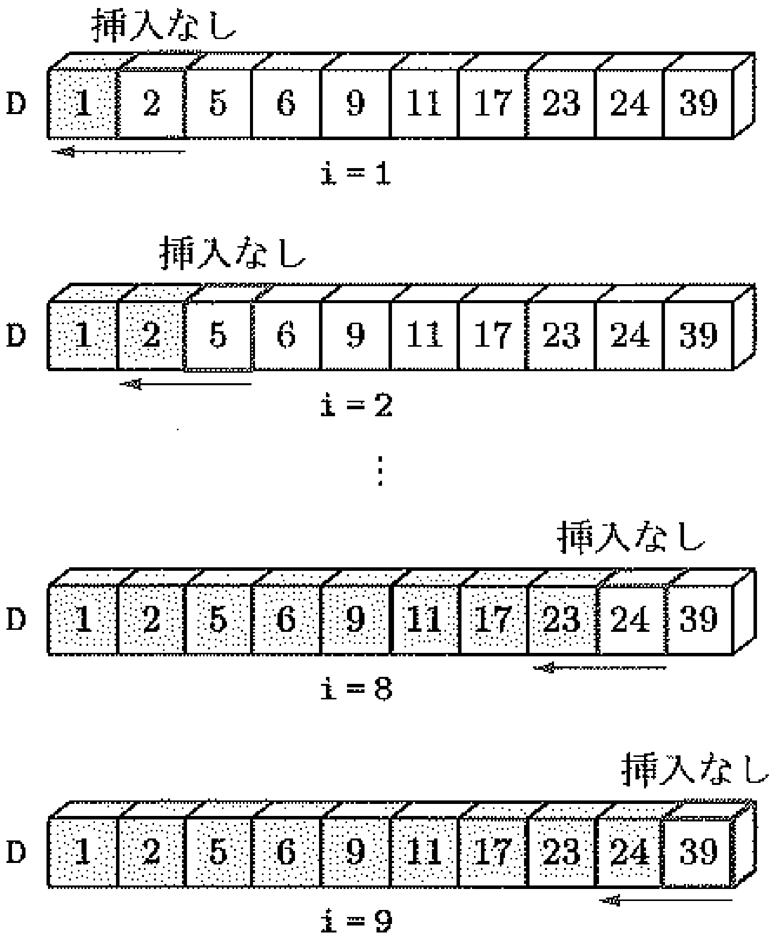


図 5.4　挿入ソートの最良の場合の実行例

計算量

最悪計算量は、降順にデータがソートされているとすると、常にデータの入れ替えが発生するため、 $O(n^2)$ となる。

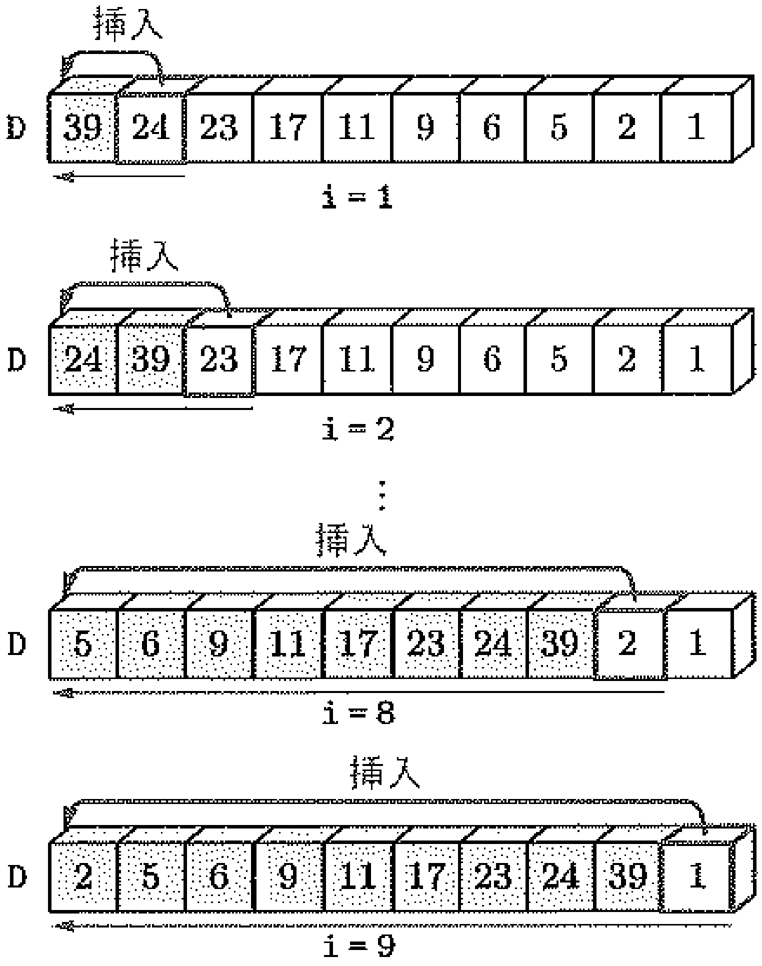


図 5.5　挿入ソートの最悪の場合の実行例

平均時間計算量

- 平均の解析は大変なため証明は略

●性質 5.1

n 個のデータに対する挿入ソートの平均時間計算量は $O(n^2)$ である．

クイックソート

- 特定の条件では高速なアルゴリズム
- 再帰処理で隣同士を入れ替えることで並べ替える

クイックソートの概念

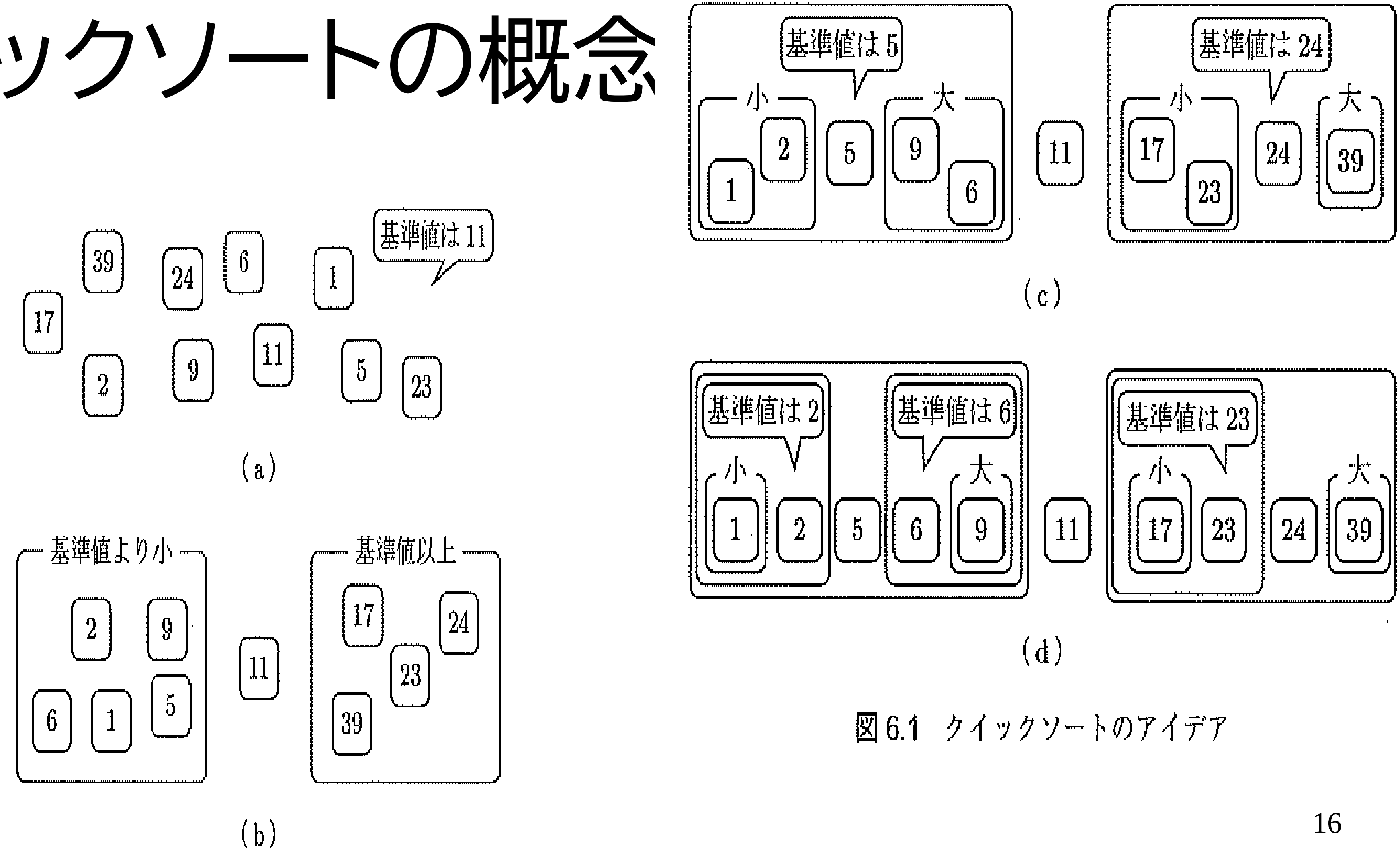


図 6.1 クイックソートのアイデア

クイックソートの手続き

$D=\{d_0, d_1, \ldots, d_{n-1}\}$ とする）.

① 集合 D に含まれる要素が 1 つならば，そのまま何もせずにアルゴリズムを終了する．

② 集合 D から適当に基準値となるデータ d_k を 1 つ選ぶ．

③ 集合 D に含まれる各データと基準値 d_k を比較し，すべてのデータをつぎのいずれかに分割する．

- d_k より小さいデータの集合 D_1
- d_k 以上のデータの集合 D_2

④ 集合 D_1 と集合 D_2 をそれぞれ再帰的にソートする．

⑤ 再帰的なソートが済んだら，3 つの集合 D_1，$\{d_k\}$，D_2 をこの順番に連結したものを出力する．

クイックソートの再帰木

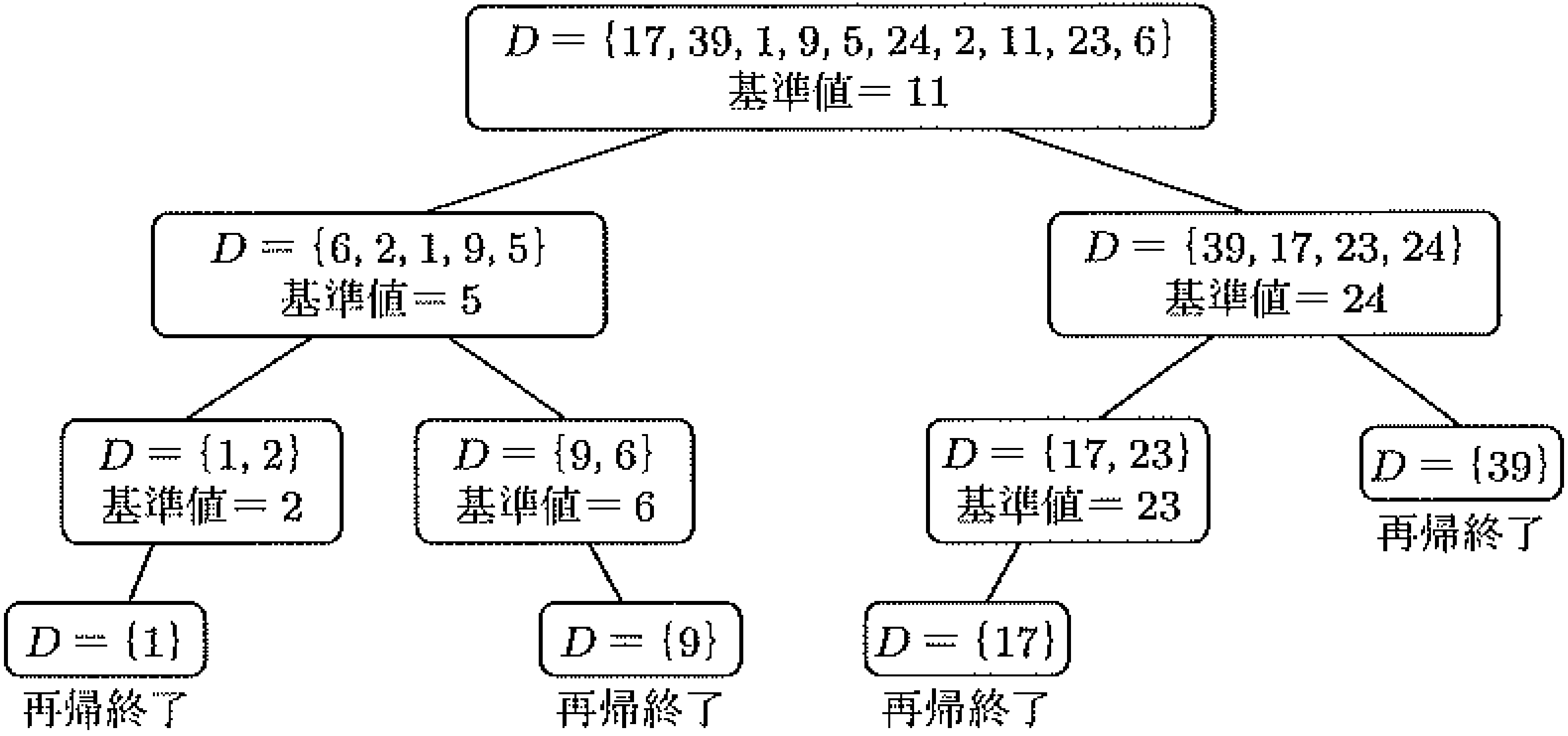


図 6.2　クイックソートの再帰木

分割の考え方

- 配列の左側に基準値より小さいデータ
- 配列の右側に基準値以上のデータ
- 上記の２つのデータの間に基準値

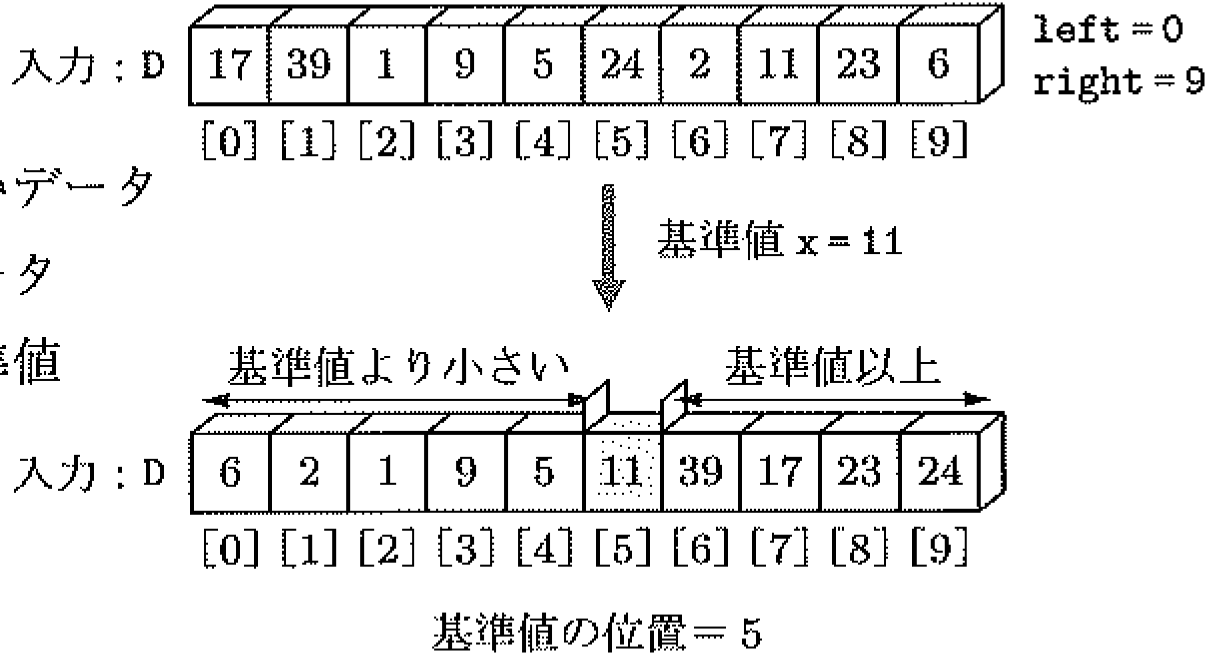


図 6.3　関数 partition の入出力例

クイックソートの手続き

アルゴリズム 6.1　クイックソート

入力：サイズ n の配列 D[0], D[1], ..., D[n-1]

```
quicksort(D,left,right) {
  if (left<right) {
    pivot_index=partition(D,left,right);
    quicksort(D,left,pivot_index-1);
    quicksort(D,pivot_index+1,right);
  }
}
```

//quicksort(D,0,n-1)|を実行することにより入力全体のソートが実行される．

分割の手順

① D[left], D[left+1], ..., D[right] の中から基準値となるデータ D[k] を選ぶ.

② 基準値 D[k] を一番右端のデータ D[right] と交換する.

③ 配列 D を D[left] から右に向かって探索し，基準値以上のデータをみつけ，その位置を変数 i に記録する.

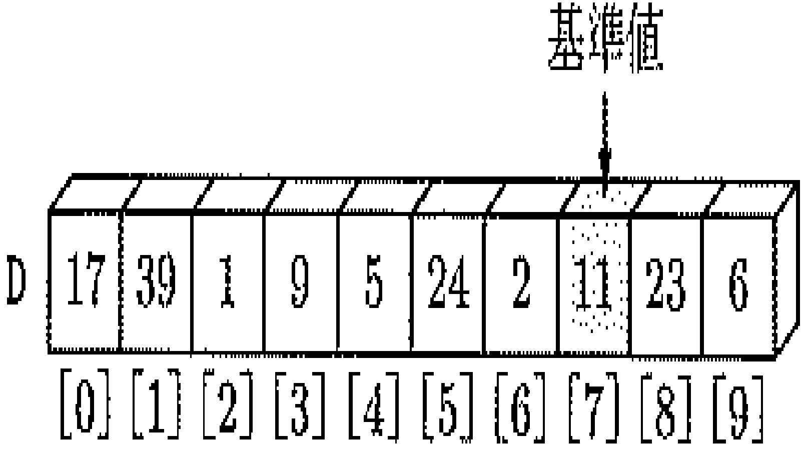
④ 配列 D を D[right-1] から左に向かって探索し，基準値より小さいデータをみつけ，その位置を変数 j に記録する.

⑤ i と j の関係が i<j であるとき，“D[i] $\geq$ 基準値 $>$ D[j]” なので，D[i] と D[j] のデータを交換する.

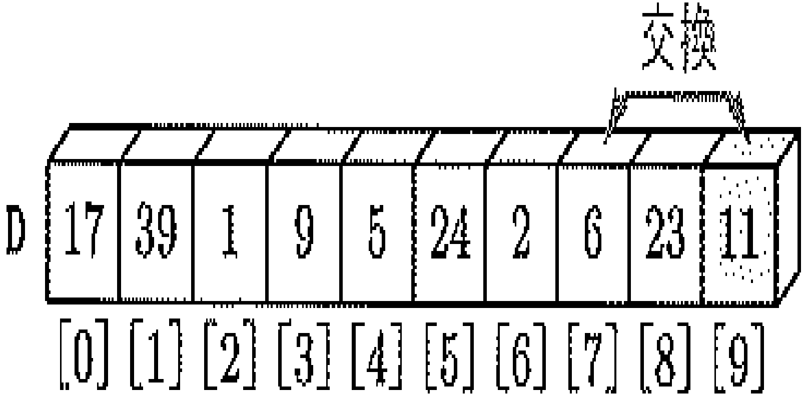
⑥ ③，④，⑤の操作を i>j となるまで繰り返す(繰り返し終了時には，基準値より小さいデータの集合と基準値以上のデータの集合に分割されている).

⑦ D[i] と D[right] のデータを交換し，基準値を 2 つの集合の間に入れる.

分割の手順


基準値
D
17 39 1 9 5 24 2 11 23 6
[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]

(a)

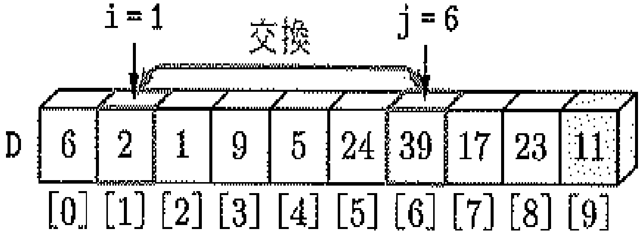

交換
D
17 39 1 9 5 24 2 6 23 11
[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]

(b)

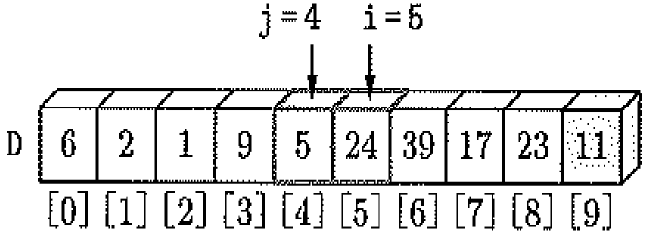
分割の手順

(e)

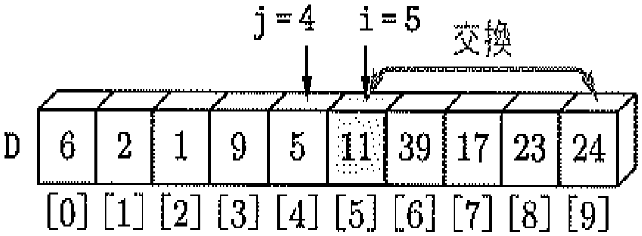
(f)


i=1
交換
j=6
D
6 2 1 9 5 24 39 17 23 11
[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]

(g)


j=4
i=5
D
6 2 1 9 5 24 39 17 23 11
[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]

(h)


j=4
i=5
交換
D
6 2 1 9 5 11 39 17 23 24
[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]

(i)

分割の手順

アルゴリズム 6.2　関数 partition

```
partition(D,left,right)  {
  基準値となるデータD[k]を選ぶ;
  swap(D[k],D[right]);        //基準値を右端のデータと交換
  i=left; j=right-1;
  while(i<j)  {
    while (D[i]<D[right]) { i=i+1; }
    while ((D[j]>=D[right])かつ(j>=i)) { j=j-1; }
    if (i<j) swap(D[i],D[j]);
  }
  swap(D[i],D[right]);        //基準値を2つの集合の間に入れる
  return i;                   //基準値の位置を出力
}
```