アルゴリズムとデータ構造

分割統治法とソート

森 立平
mori@c.titech.ac.jp

2019 年 6 月 21 日

今日のメッセージ

- **分割統治法 =「漸化式作ってそれをプログラムにするだけ」**
- **ソートは分割統治法で解ける**
- **分割統治法の時間計算量は漸化式を立てることによって見積もる**

今日の目標

- いろいろなソートアルゴリズムを習得する

1 分割統治法

「アルゴリズム ≈ 漸化式」に従って漸化式をそのままアルゴリズムにしてしまう方法を「分割統治法 (divide and conquer)」という。ユークリッドの互除法や二分探索は分割統治法の特別な場合となる (漸化式の右辺に一つしか、今計算しようとしている関数が登場しないので「decrease and conquer」と呼ばれることもあるようだ)。分割統治法が使える代表的な問題にソート問題がある。

2 ソート問題

与えられた数列 $a_1, a_2, \ldots, a_n$ を小さい順に並び変える操作のことをソートと呼ぶ。

入力： $a_1, a_2, \ldots, a_n$
出力： $a_{i_1}, a_{i_2}, \ldots, a_{i_n}$　　ここで $i_1, \ldots, i_n \in \{1, 2, \ldots, n\}$ は互いに相異なり、$a_{i_j} \leq a_{i_{j+1}}$ を満たす

このソートを計算するために分割統治法を使うことができる。だが分割統治法を考える前に、素朴な方法でどれくらいの時間計算量があるかを考えてみよう。

3 選択ソート

$$\mathrm{Ssort}(A) = \begin{cases} [\,], & \text{if } |A| = 0 \\ [\min(A)] \circ \mathrm{Ssort}(A \setminus \min(A)), & \text{otherwise.} \end{cases}$$

ここで $\circ$ は配列の連結とする。

```
void Ssort(int A[], int n){
  int i, j, min, z;
  for(i = 0; i < n - 1; i++){
    min = i;
    for(j = i+1; j < n; j++){
      if(A[j] < A[min]){
        min = j;
      }
    }
    z = A[i];
    A[i] = A[min];
    A[min] = z;
  }
}
```

4 挿入ソート

$$\mathrm{Isort}(A) = \begin{cases} [\,], & \text{if } |A| = 0 \\ \mathrm{insert}(\mathrm{last}(A), \mathrm{Isort}(A \setminus \mathrm{last}(A))), & \text{otherwise.} \end{cases}$$

ここで $\mathrm{last}(A)$ は A の最後の要素、$\mathrm{insert}(a, B)$ は、整数 a をソート済み配列 B の適切な位置に挿入することで得られるソートされた配列である。

```
void Isort(int A[], int n){
  int i, j, k, z;
  for(i = 1; i < n; i++){
    for(j = 0; j < i && A[j] < A[i]; j++) ; //二分探索で置き換え可能
    z = A[i];
    for(k = i; k > j; k--) A[k] = A[k-1];
    A[j] = z;
  }
}
```

5 時間計算量の見積り

分割統治法の時間計算量は漸化式を立てることによって見積もる。ソートの計算量を「比較の回数と代入の回数の和」と定義する。選択ソートと挿入ソートの時間計算量 $T(n)$ は漸化式

$$T(n) = T(n-1) + n$$

を満たす。よって

$$T(n) = n + (n-1) + (n-2) + \cdots + 1 = O(n^2)$$

が得られる。また、比較の回数 $U(n)$ だけを数えると、二分探索を使う挿入ソートについては

$$U(n) = U(n-1) + \log n$$

となるので、

$$U(n) = \log n + \log(n-1) + \log(n-2) + \cdots + \log 1 = \log n! \leq \log n^n = O(n \log n)$$

となる (選択ソートと二分探索を使わない挿入ソートについては $U(n) = O(n^2)$ である)。

6 マージソート

マージソートは挿入ソートと良く似ているが、配列を半分のサイズに分割する。

$$\mathrm{Msort}([a_1,\ldots,a_n]) = \begin{cases} [\,], & \text{if } n=0 \\ \mathrm{merge}(\mathrm{Msort}([a_1,\ldots,a_{\lfloor n/2\rfloor}]),\ \mathrm{Msort}([a_{\lfloor n/2\rfloor+1},\ldots,a_n])), & \text{otherwise.} \end{cases}$$

ここで $\mathrm{merge}(A,B)$ は 2 つのソート済みの配列 A と B について、$A \circ B$ をソートした配列である。

```
int B[N];

void Msort(int A[], int n){
  int i, j, k;
  if(n <= 1) return;
  Msort(A, n/2);
  Msort(A+n/2, n - n/2);
  i = j = k = 0;
  while(i < n/2 && j < n - n/2){
    if(A[i] < A[n/2 + j]) B[k++] = A[i++];
    else B[k++] = A[n/2 + j++];
  }
  while(i < n/2) B[k++] = A[i++];
  while(j < n - n/2) B[k++] = A[n/2 + j++];
  for(i = 0; i < n; i++) A[i] = B[i];
}
```

7 時間計算量の見積り

簡単のために、n を 2 の冪と仮定するとマージソートの時間計算量 $T(n)$ は

$$T(n) = 2T(n/2) + n$$

を満たす。よってこの漸化式を解くと

$$\frac{T(n)}{n} = \frac{T(n/2)}{n/2} + 1 = T(1) + \log n$$

となり $T(n) = O(n\log n)$ であることが分かる。

8 比較に基づくソートの下界と基数ソート

T 回要素の比較をして、その結果だけを使って要素の置換をしてソート問題を解いたとしよう。その場合、$2^T \geq n!$ が成り立つ。$n! \geq \left(\frac{n}{e}\right)^n$ より、$T \geq n\log n - n\log e$ が得られる。よって、比較に基づくソートでは $O(n\log n)$ という時間計算量を改善することはできない。よって、二分探索を用いた挿入ソートやマージソートの比較回数は最適であることが分かる。挿入ソートは代入の回数が $O(n^2)$ であるため時間計算量は最適ではないが、マージソートの時間計算量は最適である。

ソートに用いる値の種類が少ない場合は比較に基づかないソートがより高速である。そのようなソートとしてバケットソートと基数ソートがある。例えばソートに用いる値が 1 から L であるとする。バケットソートの概念を説明する。まず、1 から L までの値に対応したバケツを用意する。ソートしたい要素を、ソートに用いられる値に従ってバケツに入れる。全ての要素をバケツに入れたら、後は 1 から L のバケツから順番に要素を取り出せばソートができている。

9　計算量の漸化式の解き方

分割統治法の時間計算量 $T(n)$ は典型的に次の形の漸化式を満たす。

$$T(n) = aT(n/b) + f(n) \tag{1}$$

ここで $a \geq 1, b > 1$ とする。仮に $f(n) = 0$ とすると、

$$T(n) = aT(n/b) = a^2(n/b^2) = \ldots = a^{\log_b n} T(1) = O(n^{\log_b a})$$

が得られる (簡単のために n は b のべきと仮定している)。

定理 1. 式 (1)について以下が成り立つ。

1. ある $\epsilon > 0$ について $f(n) = O(n^{\log_b a - \epsilon})$ のとき、$T(n) = O(n^{\log_b a})$.
2. $f(n) = O(n^{\log_b a})$ のとき、$T(n) = O(n^{\log_b a} \log n)$.

Proof. 簡単のために n は b のべきと仮定する。まず 1. について考える。

$$\begin{aligned}
\frac{T(n)}{n^{\log_b a}} &= \frac{T(n/b)}{(n/b)^{\log_b a}} + \frac{f(n)}{n^{\log_b a}} \\
&\leq \frac{T(n/b)}{(n/b)^{\log_b a}} + cn^{-\epsilon} \\
&\leq T(1) + cn^{-\epsilon} \frac{1}{1 - b^{\epsilon}}.
\end{aligned}$$

よって $T(n) = O(n^{\log_b a})$. 次に 2. について考える。

$$\begin{aligned}
\frac{T(n)}{n^{\log_b a}} &= \frac{T(n/b)}{(n/b)^{\log_b a}} + \frac{f(n)}{n^{\log_b a}} \\
&\leq \frac{T(n/b)}{(n/b)^{\log_b a}} + c \\
&= T(1) + c \log_b n.
\end{aligned}$$

よって $T(n) = O(n^{\log_b a} \log n)$. □

基本的に分割統治法の時間計算量はこの定理を使えば求めることができる。