アルゴリズム及び演習 第 6 回演習解答

小野 孝男*

2007 年 5 月 28 日

1. 昇順にソートされた n 個のデータをマージソートでソートするために必要な比較回数を $T(n)$ とする. 明らかに $T(1) = 0$ である. また, $n \geq 2$ のときには次のように評価することができる:
   (a) 前半と後半にわけたデータのそれぞれをソートする. どちらもやはり昇順にソートされたデータであり, 従ってこのソート (2 回) に必要な比較回数は $2T(n/2)$ である.
   (b) マージ操作は, 前半にあるどのデータも後半のいずれのデータより小さいことを確認する作業となる. つまり $n/2$ 回の比較が必要である.

   従って, 次の漸化式が得られる:

$$T(n) = \begin{cases} 2T(n/2) + n/2 & n \geq 2 \text{ のとき} \\ 0 & n = 1 \text{ のとき.} \end{cases}$$

   この漸化式を解くため, 逐次代入してゆくと

$$\begin{aligned} T(n) &= 2T(n/2) + n/2 \\ &= 2[2T(n/4) + n/4] + n/2 = 4T(n/4) + n \\ &= 4[2T(n/8) + n/8] + n = 8T(n/8) + 3n/2 \\ &= \dots \end{aligned}$$

   となり, $0 \leq i \leq k$ に対して $T(2^k) = 2^i T(2^{k-i}) + i2^k/2$ と予想される. これは次のように証明できる:
   - $i = 0$ のときは 右辺 $= 2^0 T(2^{k-0}) + 0 \cdot 2^k/2 = T(n)$ より正しい.
   - $i = K$ で正しいとすると

$$\begin{aligned} T(n) &= 2^K T(2^{k-K}) + K2^k/2 \\ &= 2^K[2T(2^{k-K}/2) + 2^{k-K}/2] + K2^k/2 \\ &= 2^{K+1} T(2^{k-(K+1)}) + (K+1)2^k/2 \end{aligned}$$

     より $i = K + 1$ でも正しい.

   そこで $i = k$ とおくと $T(2^k) = 2^k T(1) + k \cdot 2^k/2 = k \cdot 2^k/2 = (n/2)\log n$ である. つまり, $(n/2)\log n$ 回の比較が必要である.

   一方, 全ての j に対し $a[j-1] \leq a[j]$ となっていることから挿入ソートでは **while** ループでの比較は各 i について 1 回しか発生しない. つまり, 挿入ソートでは $n - 1$ 回の比較が必要である.
2. 個別に考える:

* ono@is.nagoya-u.ac.jp

- **マージソート**: 再帰的に呼出すために長さが半分の 2 個の部分列を作るときに「前半」と「後半」に分割し, かつマージするときに「同じ値の要素は前半のリストにあるものを優先する」ことで安定なマージソートを作ることができる.
- **クイックソート**: 安定ではない. 例えば $\langle 3,4,2,1,2\rangle$ という列を考えると, 最初のピボット操作により $\langle 1,2,2,3,4\rangle$ という列が得られるが, この時点で 2 個の 2 の順序が交代している. そのあとで部分列 $\langle 1,2,2\rangle$ 及び $\langle 4\rangle$ を再帰的にソートするが, どちらも既にソートされているのでこれ以上変更されることなくソートは終了する. つまり, 2 個の 2 の位置が入力と出力で異なっている.
- **ヒープソート**: 安定ではない. 例えば, $\langle 2,2,2\rangle$ という列をソートすることを考える. この列はそのままヒープとなり, ヒープソートの最初の段階で先頭の 2 と最後の 2 の位置が入れ替わることになる. これで入力では先頭にあった要素が最後に移り, これ以降の処理でこの要素の位置が変わることはない.

3. 図 1 を参考に考える.

   s, t を固定する. 全ての $t' < t$ に対し $c_{st'} < c_{st}$ であることを示したい. そこで, 各 i に対し要素 c_{it} は B において位置 (s_i, t) にあった, つまり $c_{it} = b_{s_i t}$ であると仮定する. ここで $t' < t$ を固定する. B は A を各行ごとにソートして得られた配列なので $b_{s_i t'} < b_{s_i t}$ を満たす. 従って, $t' < t$ なら B の t' 列目に $b_{s_s t}$ より小さな要素は少なくとも s 個存在する. よって B を列ごとにソートした C においても t' 列目に $c_{st} = b_{s_s t}$ より小さな要素は少なくとも s 個存在することになるが, このことから $c_{st'} < c_{st}$ でなければならない (逆に $c_{s't} > c_{st}$ とすると, 昇順にソートしたことから B において t' 列目に $b_{s_s t}$ より小さな要素は高々 $s-1$ 個しか存在しないことになる). つまり C においても行ごとにソートされていることがわかる.

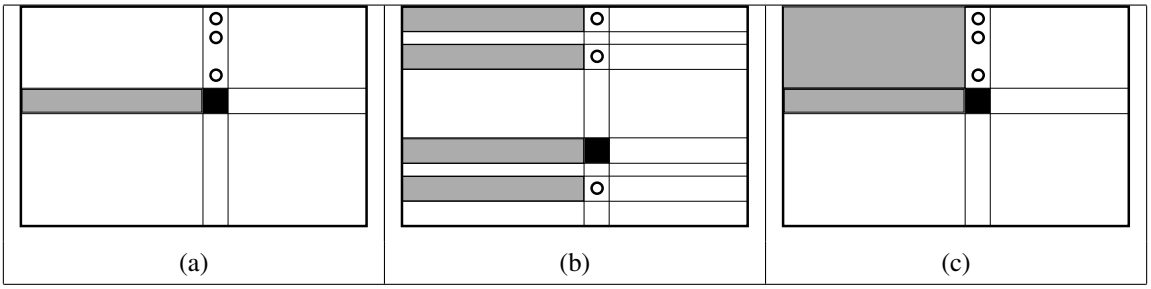

図 1　配列のソート. (a) において黒くぬった要素が灰色の要素より大きいことを示す. 黒丸の要素は黒塗りの要素より小さい. (b): (a) における黒丸の要素が図に示した場所にあったと仮定する. 灰色の要素は全て黒塗りの要素より小さい. (c): つまり灰色の要素は全て黒塗りの要素より小さい.