アルゴリズム及び演習 第 8 回演習解答

小野 孝男*

2007 年 6 月 11 日

1. 一般性を失うことなく $x_1 \le x_2 \le \cdots \le x_n$ としてよい.

   まず $x_1, x_2, \ldots, x_n$ が全て相異なる, つまり $x_1 < x_2 < \cdots < x_n$ を仮定する. このとき, 以下のように区間 I_k $(0 \le k \le n)$ を定義する:

$$I_k = \begin{cases} (-\infty, x_1) & k = 0 \text{ のとき} \\ (x_k, x_{k+1}) & 0 < k < n \text{ のとき} \\ (x_n, \infty) & k = n \text{ のとき.} \end{cases}$$

   また, $f_i(x) = |x - x_i|$ とおく. ここで, $x \in I_k$ ならば

$$f(x) = \begin{cases} x - x_i & k \ge i \text{ のとき} \\ x_i - x & k < i \text{ のとき} \end{cases}$$

   である. 従って $x \in I_k$ に対して

$$f(x) = \sum_{i \le k} (x - x_i) + \sum_{i > k} (x_i - x) = (2k - n)x + \left(\sum_{i > k} x_i - \sum_{i \le k} x_i \right)$$

   である. つまり, x の係数は x の増加とともに増加し $2k < n$ なら負, $2k > n$ なら正である. さらに, 全ての $f_i(x)$ が連続なのでその和である $f(x)$ も連続である. このことから以下のようにいうことができる:

   - n が偶数のときには $n = 2k$ とおくと $x_k \le x \le x_{k+1}$ で $f(x)$ の値は一定であり, $x < x_k$ では $f(x)$ は減少し, $x > x_k$ では $f(x)$ は増加する. つまり $x_k \le x \le x_{k+1}$ で $f(x)$ は最小値をとる.
   - n が奇数のときは $n = 2k + 1$ とおくと $x < x_{k+1}$ で $f(x)$ は減少し, $x > x_{k+1}$ なら $f(x)$ は増加する. つまり $x = x_{k+1}$ で $f(x)$ は最小値をとる.

   いずれにおいても, $x_1, x_2, \ldots, x_n$ の中央値で $f(x)$ は最小値をとる.

   次に, $x_1, x_2, \ldots, x_n$ に同じ値が存在する場合を考える. この場合も上と同じように考える. 但し, 例えば $x_k = x_{k+1}$ なら $I_k = \emptyset$ とする. このようにおけば, やはり上と同じように x の係数は x の増加とともに増加し $2k < n$ なら負, $2k > n$ なら正となる. そこで, 次のように場合分けする:

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

- $n = 2k$ が偶数で $I_k \neq \emptyset$ のとき. このとき, やはり $x \in I_k \cup \{x_k, x_{k+1}\}$ で $f(x)$ は最小値をとる.
- その他のとき. $2k < n$ かつ $I_k \neq \emptyset$ であるような最大の k を m, $2k > n$ かつ $I_k \neq \emptyset$ であるような最小の k を p とおく. この置き方から, $m + 1 = p$ であるか $x_{m+1} = x_p$ である. 前者ならば x_p は中央値であり, 後者ならば $2m < n < 2p$ よりやはり $x_{m+1} = x_p$ は中央値である.

従って, やはり $x_1, x_2, \ldots, x_n$ の中央値で $f(x)$ は最小値をとる.

2. X から k 番目に小さい要素を求めるアルゴリズムは次のように書くことができる:
   1: X のデータを 5 個ずつ組にし, 各組で中央値を求める.
   2: 1 で得られた全ての中央値から, 再帰的に中央値 x_{m} を求める.
   3: x_{m} が全体の中で何番目にあるかを調べる. m 番目であったとする.
   4: $m = k$ なら終了である. $m < k$ なら, 「x_{m} より大きなデータ」の中から $k - m$ 番目に小さい要素を求める. $m > k$ なら, 「x_{m} より小さなデータ」の中から k 番目に小さい要素を求める.

   このアルゴリズムでは, 1, 3 は $O(n)$ 時間で実行できる. 再帰呼び出しは 2 と 4 で行っているが, 対象となるデータは合計しても高々 cn 個 ($c < 1$) である. 従って全体でも $O(n)$ 時間で実行できる.

   このアルゴリズムを重み付きに拡張するときには, 「x 以下のデータの重みの和が p 以上となるような最小の x」を求めるようにする. このとき, 3 で「何番目の値か」を求める代わりに「それより小さなデータの重みの和 p'」を求めることにし, 4 では $p' \geq p$ かどうかで場合分けして再帰呼出しをすればよい.

   このように変更しても, 実行時間は本質的に変化しない. 1, 2 は全く同じであり, 3 でも元々全データに対して x_{m} と比較しなければならないので, $x_i \leq x_{\mathrm{m}}$ のときに重み w_i を加えるという操作を追加すればよく, そのためにかかる時間は $O(n)$ のままである. 最後に 4 の再帰呼び出しも全く変わらないので, 重みなしの場合に対するアルゴリズムと同じように解析することができ, 実行時間は $O(n)$ である.

3. 0 以上 $n^2 - 1$ 以下の整数は, 「n 進 2 桁」とみなすことができる. このように表現しておき, 基数ソートを実行する.

   m 進 d 桁であるような n 個の数を基数ソートでソートするには $O(d(m + n))$ 時間かかる. つまり, 「m 個のリストを初期化し, n 個のデータをその中の適切なリストに追加する」という処理を d 回繰り返すからである. 今の場合は $m = n$ かつ $d = 2$ なのでかかる時間は $O(2(n + n)) = O(n)$ である.

4. $X = \{x_1, \ldots, x_n\}$ から全ての k 分位点を求めることを考える. 全ての i に対し $\textsc{select}(x, \lfloor in/k \rfloor)$ を実行すればよいが, これは $O(nk)$ 時間かかる. これを $O(n \log k)$ 時間に削減するため, クイックソートの考え方を応用する.

   つまり, ある部分配列 $x[h, \ldots, t]$ から k 分位点 $q_i, q_{i+1}, \ldots, q_j$ を求めるため, まず これらの k 分位点のほぼ中間にある k 分位点 q_m を求め, 次に x を q_m より小さいものと q_m より大き

いものに分類する. 前者から $q_i, \ldots, q_{m-1}$ を, 後者から $q_{m+1}, \ldots, q_j$ を, それぞれ再帰的に求めればよい. このとき, クイックソートで用いた `partition` を使うことができる.

以上のように k 分位点を求める関数を $\texttt{quantile}(x[1,\ldots,n], k)$ とする. これは, 補助関数 $\texttt{quantile1}(x[1,\ldots,m], n, k, first, last, base)$ を $\texttt{quantile1}(x, n, k, 0, k, 0)$ という形で呼び出す. この関数 `quantile1` は, 元の配列においては $x[base+1]$〜$x[base+m]$ にあたる部分配列 $x[1]$〜$x[m]$ から, 元の配列に対する $q_{first+1}$〜q_{last-1} を求めるものである.

このアルゴリズムは, 本質的に「ピボットとして常に中央値をとるクイックソート」と同じ動作をする. つまり, 再帰の深さが i であるような再帰呼び出しにおいては, 調べる配列の長さは $n/2^{i-1}$ を越えない. そして, 再帰の深さは高々 $\log k + 1$ となるので実行時間の合計は $O(n \log k)$ となる.

1: **procedure** QUANTILE1(x, n, k, *first*, *last*, *base*)
2: $diff \leftarrow last - first$
3: **if** $diff < 2$ **then**
4: **return** $\emptyset$
5: **end if**
6: $mid \leftarrow \lfloor diff/2 \rfloor + first$
7: $rmid \leftarrow \lfloor n \times mid/k \rfloor$
8: $q \leftarrow$ SELECT(x, $rmid - base$)
9: partition x by pivot q so that those less than q come front.
10: **return** the union of $\{q\}$, QUANTILE1($x[1,\ldots,rmid-1], n, k, first, mid, base$), and QUANTILE1($x[rmid-base+1,\ldots,m], n, k, mid, last, rmid+1$).
11: **end procedure**