リストの可逆分割アルゴリズムを利用した ゴミ情報が最適な可逆クイック整列法の生成

2010SE273 山下健太
指導教員：横山哲郎

1 はじめに

本論文では文献 [1] におけるゴミ情報の生成・マージの手法を応用し，より一般性の高い手法を提案する．文献 [1] では，ゴミ情報量が最適なリストの可逆分割アルゴリズムを提案している．ここではさらに提案したアルゴリズムを利用してクイックソートを生成し，分割の際のゴミ情報量が最適であること，再帰呼び出しの際にゴミ情報の最適な伝播が実現できていることの 2 点を示している．しかし，この手法はクイックソートに特有のサブプロシージャに対して最適なゴミ情報の生成を保証することにより実現されており，この手法をそのまま他のアルゴリズムに応用することが難しい．そこで，本研究ではアルゴリズムのより小さな構成要素に最適なゴミ情報を生成させ，それらの最適性を保ったままより大きな構成要素へとマージを行う手法を提案し，既存の手法の応用性を高める．なお，ここではゴミ情報とは非可逆アルゴリズムを可逆化する際に記録する必要があり，可逆化する前の非可逆アルゴリズムにおいては記録する必要のない情報を表す．

2 関連研究

2.1 可逆アルゴリズム

可逆アルゴリズムとは，出力から入力を復号できるアルゴリズムである．なお，この性質は可逆性と呼ばれる．本来は可逆性をもたないアルゴリズムについても，入力を特定するための情報を出力に追加することで可逆化を行うことができる．なお，この情報はゴミ情報と呼ばれる．

2.2 In-Place アルゴリズム

In-Place アルゴリズムとは，入力されたデータを直接改変するアルゴリズムである．このようなアルゴリズムを可逆化するためには，ゴミ情報と呼ばれる付加情報を出力する必要がある．例えば n 個の要素をもつリストをソートする In-Place アルゴリズムを可逆化する場合，$\log_2 n!$ ビットのゴミ情報を出力する必要がある [2] [3].

2.3 比較ソートの理論限界

各要素の大小関係を比較し，要素の位置を入れ替えることでソートを行うソート法は比較ソートと総称される．比較ソートは入力されるデータに何らかの制約を設けない限り，入力の要素数 n 個に対して最悪計算量が $\mathrm{O}(n\log_2 n)$ を下回ることはないことが知られている．

3 既存の可逆クイックソート

文献 [1] では，従来のリストの分割操作に代わり，次のアルゴリズム $RSplit\_F(L)$ を提案している．また，$RSplit\_F(L)$ の検証を行うために，このアルゴリズムを利用したクイックソートのアルゴリズム $Q'(L)$ を示している．なお，文献 [1] では各アルゴリズムの入出力に labeled partial order を利用しているが，本論文の範囲においては通常のリストを利用しても一般性が失われないため，本論文においては通常のリストを利用している．

Algorithm 1 Reversible Split algorithm: Forward computing （[1] より改変）

Input: a discrete list L.
Output: a three-layered series list (Y_1, Y_2, Y_3) **and reversal index :** $RIndex$

$RSplit\_F(L)$:
$(Y_1, Y_2, Y_3) \leftarrow Split(L)$
$RIndex \leftarrow f(\{pos(y_i) - 1\}_{y_i \in Y_1})$
return $(Y_1, Y_2, Y_3), RIndex$

Algorithm 2 Reversible Quicksort algorithm: Forward computing （[1] より改変）

Input: a discrete list L
Output: a linear list L^* **and** *reversal index*

$Q'(L)$:
if $|L| \leq 1$ **then**
  return $(L, 1)$
else
  $(Y_1, Y_2, Y_3), C_0 \leftarrow RSplit\_F(L)$
  $(Y_1, C_1) \leftarrow Q'(Y_1)$, $(Y_3, C_2) \leftarrow Q'(Y_3)$
  return $([Y_1 : Y_2 : Y_3], |Y_3|(|L| - 1)! + C_0|Y_1|!|Y_3|! + (C_1 - 1)|Y_3|! + C_2)$
end if

$RSplit\_F(L)$ はリスト L を入力とし，L の先頭の要素をピボットとして L をサブリスト Y_1，Y_2，Y_3 に分割する．その際にピボットより大きい要素を Y_1 に，ピボットを Y_2 に，ピボットより小さい要素を Y_3 に格納する．分割後，Y_1 に格納された要素の L における位置 $\{x_i\}_{i=1}^{|Y_1|}$ を関数

$$f(\{x_i\}_{i=1}^{k}) = \sum_{i=1}^{k} x_i - 1\mathrm{C}_i \tag{1}$$

によって *RIndex* に変換し，これを記録する．

文献 [1] では $RSplit\_F(L)$ を利用してリストの分割を行うことで，可逆クイックソート $Q^{'}(L)$ を実現している．可逆性を実現するために，$Q^{'}(L)$ は 4 つの整数 $|Y_3|$，C_0，C_1，C_2 を記録する必要がある．これらはそれぞれ Y_3 の要素数，*RIndex*，Y_1 のゴミ情報，Y_3 のゴミ情報を表す．これらの情報を 4 つの値として記録する場合，$Q^{'}(L)$ が自身を再帰的に呼び出すたびに返り値の数が大きくなる．そのため，$Q^{'}(L)$ では式

$$\begin{aligned} \textit{reversal index} =& |Y_3|(|L|-1)! + C_0|Y_1|!|Y_3|! \\ &+ (C_1-1)|Y_3|! + C_2 \end{aligned} \tag{2}$$

を用いて 4 つの値から *reversal index* を算出し，これを記録する．文献 [1] において，$\{pos(y_i)-1\}_{y_i \in Y_1}$ と *RIndex* の値の組み合わせが全単射の関係にあることが証明されているため，*reversal index* とリスト (Y_1, Y_2, Y_3) の組み合わせからリスト L を一意に特定できることが分かっている．また，要素数が n 個のリスト L に対し *RIndex* の値が $n!$ 通りの値をとることから，$Q^{'}(L)$ の処理全体ではゴミ情報量が最適であることが明らかである．そのため，既存の手法によって 4 つの情報を最適にマージできていることが分かっている．しかし，文献 [1] ではその理由が明らかにされていない．また，文献 [1] で示されている手法は 4 つの整数をマージするものであり，一般性が低い．そのため，現時点では文献 [1] の手法を他のアルゴリズムに応用することが難しい．

4 提案する手法

4.1 ゴミ情報の定式化と演算子 ⋈

そこで，本研究ではゴミ情報を 2 つの整数の組として定式化し，2 つのゴミ情報をオペランドとして *reversal index* を求める演算子 ⋈ を提案する．演算子 ⋈ によって行われる演算は，式

$$(g_1, h_1) \bowtie (g_2, h_2) = (g_1 \times h_2 + g_2, h_1 \times h_2) \tag{3}$$

として定式化できる．ただし，n は自然数，g_n はゴミ情報の値，h_n は g_n の取り得る最大値に 1 を加算した値とする．また，ここでは演算子 ⋈ は結合性をもつ．演算子 ⋈ を用いて，式 (1) と同様の *reversal index* を生成可能であることが確認できる．既存の手法を利用して *reversal index* を生成する際には不必要な情報が発生していないことが証明されているため，⋈ によってゴミ情報の最適なマージができていることが確認できる．

4.2 プロセスを細分化した可逆分割アルゴリズム

可逆アルゴリズムの一般性を高めるため，Algorithm 1 に対して処理の細分化を行ったアルゴリズムが次に示す Algorithm 3 である．Algorithm 3 は入力としてリスト L，サブリストの組 (Y_1, Y_2, Y_3)，ゴミ情報 *RIndex* を受け取り，リスト L の先頭の要素を適切なサブリストに格納する．このとき要素を Y_1 に格納した場合はゴミ情報 ${}_{|Y_1|+|Y_3|-1}\mathrm{C}_{|Y_1|}$ を生成し，その時点での *RIndex* に加算することでマージを行う．以上の処理を再帰的に繰り返すことで，Algorithm 3 は Algorithm 1 と同様の分割処理を行う．

Algorithm 3 Reversible Split algorithm: Forward computing （Algorithm 1 より改変）

Input: a discrete list L, **a three-layered series list** (Y_1, Y_2, Y_3), **and reversal index :** *RIndex*.
Output: a three-layered series list (Y_1, Y_2, Y_3), **and reversal index :** *RIndex*.

$RSplit\_F^{'}(L, (Y_1, Y_2, Y_3), RIndex)$:
if $Y_2 = \emptyset$ **then**
  $Y_2 \cup L[0]$, $L \leftarrow tail(L)$
else if $L[0] < Y_2[0]$ **then**
  $Y_1 \cup L[0]$, $L \leftarrow tail(L)$
  $RIndex \leftarrow RIndex + {}_{|Y_1|+|Y_3|-1}\mathrm{C}_{|Y_1|}$
else
  $Y_3 \cup L[0]$, $L \leftarrow tail(L)$
end if
if $L \neq \emptyset$ **then**
  $((Y_1, Y_2, Y_3), RIndex) \leftarrow RSplit\_F^{'}(L, (Y_1, Y_2, Y_3), RIndex)$
end if
return $((Y_1, Y_2, Y_3), RIndex)$

5 おわりに

本研究では，ゴミ情報の定式化・ゴミ情報のマージを行う演算子 ⋈ の定義を行った．これによって，プロセスを細分化した可逆クイックソート・可逆マージソートを実現した．今後の課題として，入力の組み合わせ以外を記録するためのゴミ情報を生成するプロセスの確立が挙げられる．これを実現できた場合，リストの分割を利用したアルゴリズム以外のアルゴリズムの可逆化が可能になると考えられる．

参考文献

[1] Early, D., Gao, A. and Schellekens, M.: Frugal Encoding in Reversible $\mathcal{MOQA}$: A Case Study for Quicksort, *Proc. Reversible Computation*, Glück, R. and Yokoyama, T. (Eds.), Lecture Notes in Computer Science, Vol. 7581, Springer-Verlag, pp. 85–96 (2012).

[2] Perumalla, K. S.: *Introduction to Reversible Computing*, CRC Press (2013).

[3] Hall, J. S.: A Reversible Instruction Set Architecture and Algorithms, *Proc. Physics and Computation*, IEEE Press, pp. 128–134 (1994).