

クリーンな可逆ハフマン木構築法の効率化および解析

M2022SE006 児玉春司

指導教員：横山 哲郎

1 はじめに

各ステップで直前と直後の状態を高々一つしかもたないアルゴリズムを可逆という。これまでに知られている効率的な可逆アルゴリズムは、長年にわたって研究が蓄積されてきた効率的なアルゴリズムの数に比べて少ない。ゼロから可逆アルゴリズムを構築するのではなく、対応するアルゴリズムを可逆化することで、これまでの研究蓄積を可逆計算で応用することが可能である（例えば、[5, 2]）。非単射な関数を単射化した関数には余剰出力がある。したがって、単射化した関数を実現する可逆アルゴリズムも余剰出力をもつ。この余剰出力は得たい結果ではないことからゴミと呼ばれる。ゴミを減らし必要な計算リソースを小さくすることは、可逆アルゴリズムの効率化に繋がる。

ハフマン符号化は、ハフマン木を用いて最適な接頭符号を生成するデータ圧縮などで広く用いられている手法である。ハフマン符号化を可逆化した田島の手法 [7, 可逆な解法 2] は一般解法 [5, 2] で得られた結果よりも効率的である。しかし、この手法の最適性は確認されておらず、正当性の評価は十分でない。

本研究の目的は、田島の手法における冗長性の特定と改良をし、効率的でクリーンな可逆ハフマン木構築法を得ることである。特に、田島の手法において 2 回行われるハフマン木の走査を 1 回に削減し、またこれに伴う処理を削減する。さらに、提案手法が、可逆計算における漸近的実行時間とゴミ出力の最適性を表す衛生性をもつことを示す。

今後の研究課題としては、可逆的にハフマン木を構築するのに必要とされる最適な比較回数を特定すること、および一般解法により得られた可逆化や他の可逆ハフマン木構築法との比較をすることがあげられる。

本研究で得られた知見により、効率的な可逆ハフマン符号化手法に関する理解が深まり、他の手法の可逆化が進展することが期待される。

2 関連研究

2.1 可逆計算

可逆計算は、計算過程において、任意の計算ステップで直前と直後の状態が高々一つに定まる計算である。ここで、計算過程において、任意の計算ステップで、現在の状態から直後の状態が高々一つに定まることを前方決定的、現在の状態から直前の状態が高々一つに定まることを後方決定的、前方かつ後方に決定的であることを可逆的という。可逆計算は出力から、入力が一意に定まる単射性をもつ。

任意の関数 f に対して、余剰な出力を表すある関数 g が

存在して、 $h(x) = (f(x), g(x))$ は単射である。ここで f から h を得ること、および h を f の単射化といい、 $g(x)$ を x に対するゴミという。実用上の目標としてゴミを減らすことは、可逆計算において時間や空間のリソースを節約するために重要である。

Landauer は、コンピュータの計算において、情報を損失したときに放熱し、エネルギーが消費されることを示した [5]。可逆計算では、情報損失がないため、消費エネルギー最適化に应用されている。

2.2 可逆アルゴリズム

アルゴリズムにおける原子的なステップが可逆とはそのステップが表す関数が単射であることをいう。可逆アルゴリズムとは、全ステップが可逆なアルゴリズムである。

関数 f を計算する（非可逆）アルゴリズム X の可逆化とは、 X から、 f を単射化した関数を計算する可逆アルゴリズムへ変換すること、およびその可逆アルゴリズムである。可逆アルゴリズムは、入力から出力を得る（順）実行に加えて、出力から入力を得る逆実行が可能である。

2.2.1 可逆アルゴリズムの指標

可逆アルゴリズムの性能を評価するためのいくつかの指標が知られている。クリーン性とは、 f の単射化関数 $h(x) = (f(x), g(x))$ における g が定数関数であることであり、ゴミ出力が無いことを意味する。

（非可逆）アルゴリズム X が計算する関数の単射化関数を計算するアルゴリズム X' を X の可逆化という。 X' が X に対して忠實的 [1] とは、ゴミ出力量の上界を定める関数 $g: \mathbb{N} \rightarrow \mathbb{N}$ に対して、次の 3 条件を満たすことをいう：

1. $g(|x|)$ の定数倍以下のゴミ出力量
2. 漸近的な時間計算量の変化なし
3. 追加の漸近的な空間計算量が最大でも $g(|x|)$

X' は、 X に対して忠實的かつゴミ出力量が最適であるとき、衛生的という [1]。

2.2.2 可逆化手法

アルゴリズムの可逆化の手法には、個別解法と一般解法がある。例えば、比較整列法に対する個別解法には、一般解法とは異なった特徴をもつものが知られている [1]、

最も単純な一般解法は、計算履歴をすべて保持する方法である [5]。この方法では、元のアルゴリズムと可逆化アルゴリズムの漸近的な時間計算量は同一であり、可逆化アルゴリズムの空間計算量とゴミ出力量は元のアルゴリズムの時間計算量に比例して増加する。

また、可逆化手法を用いて得られた元の出力を保存した後に、それを逆実行することでゴミ出力量を削減する方法が知られている [2]。この逆実行は、順実行で結果を得た際に発生したゴミを消すために行う。このとき、元の入力 がゴミとなる。

2.3 高水準可逆オブジェクト指向言語 ROOPL

可逆アルゴリズムの記述には高水準の可逆プログラミング言語が適している。可逆言語は全ての原子ステップにおいて単射であることが保証されており、そのプログラムは可逆であることが保証される。

本研究では、高水準可逆オブジェクト指向言語 ROOPL を用いてプログラムを記述する。ROOPL の特徴的な構文を簡潔に以下に示す。

`local t x = e` は、型 t で初期値が式 e の評価値であるストレージをアロケートしてローカル変数 x にその参照を格納して環境に x を追加する。`local` で定義した変数は `delocal` を用いて消去する必要がある。`call/uncall` 文は、メソッド呼び出しをする。`call` 文はメソッドを順実行し、`uncall` 文はメソッドを逆実行する。`new/delete` 文は、オブジェクトを生成/消去する。`delete` 文で消去されるオブジェクトはゼロクリアされている必要がある。

2.4 ハフマン符号化

ハフマン符号化とは、出現数が多い文字に対して短い符号語を割り当て、可変長符号を設計する手法である。ハフマン符号化には、葉に出現文字と頻度をもつハフマン木を用いてハフマン符号を生成する。ハフマン符号化によって得られる可変長符号は、復号の際に一意に復元可能である。

Leeuwen のアルゴリズム A [6] は、文字と頻度の組を頻度順に昇順に整列した入力列 IN からハフマン木を構築する手法である。ハフマン木は、各葉は文字と文字の頻度をもち、各節はすべての子孫の葉の文字の頻度の合計をもつ。アルゴリズム A は、 IN が空で Q のサイズが 1 になるまで以下の処理を繰り返す。ただし、ハフマン木の部分木の列を一時的に保存するのにキュー Q を用いる。

1. IN の先頭から 2 つの組、 IN と Q の先頭からそれぞれ 1 つの組、および Q の先頭から 2 つの組のうちから頻度の和が最小となるものを α と β とする。
2. 新しい節 γ を作り、その頻度を α と β の頻度の和とし、その左右の子どもを α と β またはその逆とする。
3. α と β を元のキューから削除する。
4. γ を Q にエンキューする。

繰返しごとに IN と Q の要素数の和が 1 減り、サイズ $\Theta(1)$ の新しい節が作成されるので、アルゴリズム A の時間計算量は $\Theta(n)$ 、入出力を除く空間計算量は $\Theta(n)$ である。

処理 2 は、左の子が α 、右の子が β になるときとその逆のときがあるので、前方決定的でない。

処理 2 は γ の左の子が α 、右の子が β であったときとその逆のときがあり、処理 3 は α と β を削除しており、共に

後方非決定的である。

したがって、アルゴリズム A は、前方決定的でも後方決定的でもなく、非可逆である。

アルゴリズム A は次のように決定的にして入力を制限することで単射になる。2 のプロセスを左右の子どもを α 、 β の順に固定することで決定的にする。

可逆キュー [7] を用い、また入力を頻度の昇順に並べられた列に限定することで、単射アルゴリズムにする。3 のプロセスで α 、 β を元のキューから削除しているが、 α 、 β をデキューすれば、削除する必要がない。

2.5 可逆ハフマン符号化

可逆ハフマン符号化は、2 つの解法が提案されている。

Hay-Schmidt [3] は、Huffman [4] の手法を優先度付きキューを用いて実装した手法を可逆化することで、可逆ハフマン符号化手法を提案した。

この手法の漸近的な時間計算量は $O(n \lg n)$ 、入出力を除く空間計算量は $O(n \lg n)$ 、ゴミ出力量は $\Theta(n \lg n)$ であり、可逆化前の時間計算量 $O(n \lg n)$ 、空間計算量 $O(n \lg n)$ となっているため、Huffman [4] のハフマン符号化手法に対して忠実な手法である。

田島 [7] は、Leeuwen [6] の手法を前方決定的にした手法 L を提案し、この手法を後方決定的にして計算過程の各ステップを可逆化することで、可逆ハフマン符号化手法を提案した。

この手法は、時間計算量 $\Theta(n)$ 、入出力を除く空間計算量 $O(n)$ 、ゴミ出力量なしであり、可逆化前の時間計算量 $\Theta(n)$ 、空間計算量 $O(n)$ であるため、手法 L に対して衛生的なアルゴリズムである。

田島の手法は、既存の方法の中で最も効率的であるが、検証が十分になされていない。

3 提案手法

田島のアルゴリズム [7] は入力をキュー IN としてハフマン木を生成し、ハフマン木の先頭を指すノード $root$ を出力するアルゴリズムである。田島のプログラムを検証した結果、冗長な部分を発見したので、冗長部分の改良を行った。

3.1 改良プログラム

田島のアルゴリズムを改良したアルゴリズムを図 1 と図 2 に示す。

IN は入力のキューであり、 Q は一時的にハフマン木の節を格納するためのキューである。キューは、先頭の要素への参照 $head$ 、要素数 $length$ をもち、エンキューとデキューをメソッドとしてもつ。 $root$ は最終的にハフマン木の根への参照を格納する。ハフマン木の $node$ は文字の頻度 w 、左の子への参照 l 、右の子への参照 r をもち、 l 、 r ともに nil であるとき、ハフマン木の葉である。

メソッド $deqMin$ はキュー IN と Q の先頭のうち小さい方をゼロクリアされた res に格納する。 $deqMin$ 内で

```

1 method huffmanTree(Queue IN, Node root)
2   local Queue Q = nil
3   new Queue Q
4
5   from Q.length = 0 loop
6     local Node node = nil
7     new Node node
8     local Node left = nil
9     local Node right = nil
10    call deqMin(IN, Q, left)
11    call deqMin(IN, Q, right)
12    node.w ^= left.w + right.w
13    node.l <=> left
14    node.r <=> right
15    call Q::enq(node)
16    delocal Node right = nil
17    delocal Node left = nil
18    delocal Node node = nil
19  until IN.length = 0
20
21  call Q::deq(root)
22  delete Queue Q
23  delocal Queue Q = nil

```

図1 改良ハフマン木構築プログラム

```

1 method deqMin(Queue IN, Queue Q, Node res)
2   local Node IN_H = nil
3   call IN::deq(IN_H)
4   local Node Q_H = nil
5   call Q::deq(Q_H)
6
7   if IN_H != nil &&
8     (Q_H = nil || IN_H.w < Q_H.w) then
9     res <=> IN_H
10    uncall Q::deq(Q_H)
11  else
12    res <=> Q_H
13    uncall IN::deq(IN_H)
14  fi res.l = nil && res.r = nil
15
16  delocal Node Q_H = nil
17  delocal Node IN_H = nil

```

図2 改良 dequeueMin プログラム

メソッド deq は call で呼び出されるとデキューを行い、uncall で呼び出されるとデキューの逆実行を行う。改良したプログラムの変更点を次節に示す。

3.2 田島のアフマン木構築プログラムからの変更点

田島のアフマン木構築プログラム (図3) では6行目の from 文から22行目まで、ループ変数を使用してループを行っている。また、26行目にアフマン木の葉の数を引数 n の値から足す leaf メソッドを uncall して、ループ後に葉の数を数え直してゼロクリアをしている。

```

1 method huffmanTree(Queue IN, Node root)
2   local int n = IN.length
3   local Queue Q = nil
4   new Queue Q
5   local int i = 0
6
7   from i = 0 loop
8     // 図1の6-18行目と同様
9     i += 1
10    until i = n - 1
11
12    delocal int i = n - 1
13    call Q::deq(root)
14    delete Queue Q
15    delocal Queue Q = nil
16    uncall root::leaf(n)
17    delocal int n = 0

```

図3 田島のアフマン木構築プログラム [7, 可逆な解法2]

改良したアフマン木構築プログラム (図1) では、ループの始めは、キュー Q の長さが0であり、アフマン木が構築されると、入力キュー IN の長さが0になることに着目して、ループ変数の代わりに事前条件と事後条件を設定することで、ループ変数を消去した。ループ変数を消去することで、leaf メソッドでゼロクリアする必要がなくなったので、leaf メソッドも消去した。

田島の dequeueMin プログラム (図4) では、deqMin メソッドについて、条件分岐を isInputLowerThanQueue という変数を用いて行っていた。

改良した dequeueMin プログラム (図2) では、条件分岐を整理し、キュー IN の先頭とキュー Q の先頭を直接比較することで冗長な文を削除した。

3.3 衛生性と1パスの証明

アルゴリズムの改良によって出力されるアフマン木の走査が2パスから1パスに削減されたことを示す。元のプログラムでは、ループ変数を用意して、0から増分して入力キューの長さになるまで繰り返しを行っていた。また、leaf メソッドを使ってアフマン木を走査していたので、2パスのアルゴリズムであった。変更後は、事前条件を一時的なキューが空であること、事後条件を入力キューが空であることと設定すると、繰り返しの途中では両方のキューが空になることはないので変更前と同様に動作する。この変更に伴って、走査回数が1回になったことに加え、ループ変数の定義、削除と入力キューの長さを格納する変数 n の定義、ゼロクリア、削除は不要になった。

田島のアルゴリズムは、Leeuwen のアフマン木構築法に対して衛生的であることが主張されているが、証明がなされていない。そのため、田島のアルゴリズムおよび改良したアルゴリズムで証明を行う必要がある。衛生性を示す前段階として、アルゴリズムの性能を以下に示す。enq, deq

```

1 method deqMin(Queue IN, Queue Q, Node res)
2   local Node IN_H = nil
3   if IN.length != 0 then
4     call IN::deq(IN_H)
5   fi IN_H != nil
6   local Node Q_H = nil
7   if Q.length != 0 then
8     call Q::deq(Q_H)
9   fi Q_H != nil
10
11   local int isInputLowerThanQueue = nil //
12     0: false, 1: true
13
14   if IN_H = nil then
15     skip
16   else if Q_H = nil then
17     isInputLowerThanQueue ^= 1
18   else if IN_H.w < Q_H.w then
19     isInputLowerThanQueue ^= 1
20   fi IN_H.w < Q_H.w
21   fi Q_H = nil
22   fi IN_H = nil
23
24   if isInputLowerThanQueue = 1 then
25     res <=> IN_H
26     if Q_H != nil then
27       uncall Q::deq(Q_H)
28     fi Q.length != 0
29     isInputLowerThanQueue ^= 1
30   else
31     res <=> Q_H
32     if IN_H != nil then
33       uncall IN::deq(IN_H)
34     fi IN.length != 0
35   fi res.left = nil && res.right = nil
36
37   delocal int isInputLowerThanQueue = 0
38   delocal Node Q_H = nil
39   delocal Node IN_H = nil

```

図4 田島の dequeueMin プログラム [7, 可逆な解法 2]

は、キューの要素数 n に対して時間計算量 $\Theta(1)$ 、入出力を除く空間計算量 $O(1)$ 、ゴミ出力量ゼロである。deqMin も、時間計算量 $\Theta(1)$ 、入出力を除く空間計算 $O(1)$ 、ゴミ出力量ゼロである。leaf は、時間計算量 $\Theta(n)$ 、入出力を除く空間計算量 $O(1)$ 、ゴミ出力量ゼロである。よって、このアルゴリズムは時間計算量 $\Theta(n)$ 、入出力を除く空間計算量 $O(n)$ 、ゴミ出力量ゼロであり、衛生的な解法である。(非可逆なアルゴリズムでは、時間計算量 $\Theta(n)$ 、入出力を除く空間計算量 $O(n)$)

可逆ハフマン符号化アルゴリズム全体に関しては、事前に昇順にしているため、ソートの計算量やゴミ出力量も考慮する必要がある。ソート方法を変えることにより、ハフマン符号化アルゴリズム全体の計算量が変化する。時間計

算量を最適化する場合は時間計算量 $O(n \lg n)$ 、ゴミ出力量 $O(n \lg n)$ であり、ゴミ出力量を最適化する場合は時間計算量 $O(n^2)$ 、ゴミ出力量 $\lceil \lg(n!) \rceil$ ビットとなる。

4 おわりに

我々の知る限りで最も効率的な既存手法である田島の可逆ハフマン木構築法を元に、その冗長部分を特定し改良を行った。この改良により、ハフマン木の走査を従来の 2 回から 1 回に削減し、それに伴う処理を削減した。また、衛生的性を示すために改良アルゴリズムの解析をした。

可逆アルゴリズムを衛生的でクリーンにするには余剰データを出力に含められない。このため、提案手法のように、走査をしなければ得られないノード数のようなデータを実行中にもたないことで走査回数を削減すること、及びループやアサーションに余剰なデータに依存しない適切な式で与えることは重要である。田島や我々が採用した入力データの制限による可逆化の効率化は応用が期待される。

既存の 2 つの手法と提案手法、別のハフマン木構築法の一般解法による可逆化とを比較することにより、可逆ハフマン木構築法の設計を一般化することが期待される。

可逆ハフマン木構築法の最適性の確認に必要な最適比較回数の特定することは今後の課題である。

参考文献

- [1] Axelsen, H. B. and Yokoyama, T.: Programming Techniques for Reversible Comparison Sorts, *Programming Languages and Systems* (Feng, X. and Park, S., eds.), Lecture Notes in Computer Science, Vol. 9458, Springer-Verlag, pp. 407–426 (2015).
- [2] Bennett, C. H.: Logical Reversibility of Computation, *IBM Journal of Research and Development*, Vol. 17, No. 6, pp. 525–532 (1973).
- [3] Hay-Schmidt, L.: Investigating the Reversibilization of Irreversible Algorithms, Master’s thesis, University of Copenhagen (2021).
- [4] Huffman, D. A.: A Method for the Construction of Minimum-Redundancy Codes, *Proceedings of the IRE*, Vol. 40, No. 9, pp. 1098–1101 (1952).
- [5] Landauer, R.: Irreversibility and Heat Generation in the Computing Process, *IBM Journal of Research and Development*, Vol. 5, No. 3, pp. 183–191 (1961).
- [6] Van Leeuwen, J.: On the construction of Huffman trees, *International Colloquium on Automata, Languages and Programming. Proceedings*, Edinburgh University Press, pp. 382–410 (1976).
- [7] 田島嘉人：可逆アルゴリズムのゴミ出力量の最適化—深さ優先探索とハフマン符号化を対象として—, 2021 年度修士論文, 南山大学 (2022).