可逆な文字列照合アルゴリズム
—力任せ法と Rabin–Karp 法を対象として—

2018SE016 平工真基　　2018SE091 谷崎海良
指導教員：横山哲郎

1　はじめに

可逆計算は単射な計算のみを行うことで発生する熱をなくしてエネルギーの消費を小さくしたり量子計算に応用したりできる可能性がある [6].

可逆計算は全ステップが単射でなければならない．しかし，既存のアルゴリズムは全ステップが単射であるとは限らない．よって，既存のアルゴリズムを可逆計算で用いるにはアルゴリズムの全ステップが単射であることを確かめ，非単射なステップが存在する場合は非単射なステップを単射なステップにする必要がある．アルゴリズムの全ステップを単射なステップにすることを可逆化という．非可逆なアルゴリズムを可逆化する方法として Landauer 法 [4] や Bennett 法 [2] が存在する．しかし，Landauer 法や Bennett 法でアルゴリズムを可逆化した場合，元の非可逆なアルゴリズムと比較して空間計算量が悪化したりゴミが発生したりするという問題がある．今までは線形探索など特定の非可逆なアルゴリズムを可逆化する方法を提案することで Landauer 法や Bennett 法で可逆化するより時間計算量などの点で優れた可逆なアルゴリズムを生み出す研究がされてきた [5].

本研究は文字列照合法のうち，基本的なアルゴリズムである力任せ法と Rabin–Karp 法の faithful な可逆なアルゴリズムの提案を目的とする．文字列照合法はファイルから特定の文字列を検索，DNA 配列の中から特定のパターンを発見など様々な場面で有用なアルゴリズムなので本研究の対象とした.

各アルゴリズムは [3] を参考にする．単射なステップと非単射なステップを明確にし，非単射なステップを単射なステップにすることで各アルゴリズムを可逆化する．可逆プログラミング言語 Janus を用いて可逆化したアルゴリズムを実装することで提案した方法によって可逆化したアルゴリズムが可逆であることを示す．可逆化した各アルゴリズムの時間計算量，空間計算量，及びゴミの量を解析し，非可逆な文字列照合法と可逆な文字列照合法の解析結果の比較及び可逆な文字列照合法同士の解析結果の比較をすることで提案したアルゴリズムを評価する.

2　準備

2 章では可逆アルゴリズムや文字列照合問題など本研究に関わることについて説明する.

2.1　可逆計算

可逆計算とは任意の計算ステップで直前の状態が高々一意に定まる計算である．任意の可逆計算は計算後の状態及び計算前の状態が高々一意に定まる．よって，可逆計算で行われる任意の計算は単射部分写像である.

2.2　可逆プログラミング言語

可逆なプログラムとはプログラムを実行するとき，任意の状態から直前の状態を高々一意に定めることができるプログラムである．可逆プログラミング言語は可逆なプログラムしか書けないプログラミング言語なので，変数に別の値を直接代入するなどの非可逆な処理はできない．可逆プログラミング言語には Janus や ROOPL などがある．Janus の構文は可逆性を保てるようにいくつかの制約や条件があり，意図せずゴミが出ない設計である．Janus はローカル変数の宣言に local，delocal を用いる．local t $n = v$ と書くことで型が t，初期値が v の変数 n を宣言でき，local を delocal にすると型が t，値が v の変数 n をゼロクリアして解放できる．if 文は if e_1 then s_1 else s_2 fi e_2 と書き e_1 が真ならば s_1 を実行し偽ならば s_2 を実行する．s_1 を実行後 e_2 は真，s_2 を実行後 e_2 は偽である必要がある．from 文は from e_1 do s_1 loop s_2 until e_2 と書く．最初 e_1 は真であり，s_2 を実行後は偽である．s_1 を実行後 e_2 が真なら繰り返しを終了し偽なら s_2 を実行する．s_2 を実行後，s_1 を実行するステップから繰り返す．call は call p と書くことでプロシージャ p を実行することができ，uncall p と書くことで p を逆実行できる.

2.3　計算量

計算量はプログラムを実行する際に時間やメモリなどの資源を消費する量を表すものである．計算量のうち時間を消費する量を表したものを時間計算量，メモリを消費する量を表したものを空間計算量という．1 番大きい項以外と定数係数を無視して考える計算量を漸近的な計算量といい，オーダー記法を用いて表す.

2.4　ゴミ

非可逆なプログラムを可逆なプログラムにすることを目的として，非可逆なプログラムでは出力しないが可逆なプログラムでは出力するものをゴミという．ゴミが多い場合，非可逆なプログラムと比べ空間計算量が大きくなる．ゴミの量の指標として faithful と hygienic がある [1]．漸近的な時間計算量が対応する非可逆なプログラム以下，空間計算量が非可逆なプログラムの漸近的な空間計算量に入力のサイズを足した量の定数係数倍以下，ゴミの量が入力のサイズを定数係数倍した量以下であるとき faithful であるという．faithful であるプログラムのうち，ゴミの量が最適であるものを hygienic であるという.

2.5 可逆アルゴリズム

可逆アルゴリズムはアルゴリズムの全てのステップが単射であるようなアルゴリズムである．全てのステップが単射なので可逆アルゴリズムでは出力から入力を一意に定めることができる．非可逆なアルゴリズムの各ステップを単射なステップのみにすることを可逆化という．非可逆なアルゴリズムを可逆化する一般解法としては Landauer 法 [4] や Bennett 法 [2] などがある．

Landauer 法ではアルゴリズムを実行するときに発生する情報を保存することで可逆化を行うので，出力するときに余分な情報がゴミとなるので空間計算量が大きくなる．

Bennett 法ではアルゴリズムが可逆になるようにゴミを生成しながら実行した後，アルゴリズムの結果のみをコピーする．その後，アルゴリズムを逆方向に実行することによってアルゴリズムを順番に実行したときに生成されたゴミを消去する．ゴミを消去することにより，最終的に入力と結果のみを残すことができるがアルゴリズムを逆方向にも実行する必要があるので非可逆なアルゴリズムと比較して時間計算量が約 2 倍になる．

2.6 文字列照合問題

テキスト t を長さ n の文字列，パターン p を長さ m の文字列とする．t，p の i 文字目は t_i，p_i とする．$0 \leq i \leq n-m$ を満たす整数 i のうち，$t_i = p_0$，$t_{i+1} = p_1$，$\cdots t_{i+m-1} = p_{m-1}$ となるような i があるとき，p は t の位置 i に存在するとする．$0 \leq i \leq n-m$，$t_i = p_0$，$t_{i+1} = p_1$，$\cdots t_{i+m-1} = p_{m-1}$ の 2 つの条件を満たす整数 i を全て求めることが文字列照合問題の目的である．

3 力任せ法による文字列照合

$0 \leq i \leq n-m$ の任意の整数 i について $t_i = p_0$，$t_{i+1} = p_1$，$\cdots t_{i+m-1} = p_{m-1}$ かを調べることで p が t のどの位置に存在するかを調べるアルゴリズムである．力任せ法はパターン，テキスト，パターンとテキストの長さが入力，パターンがテキストに存在する位置の列が出力である．力任せ法の時間計算量は $\mathrm{O}(m(n-m+1))$ であり，空間計算量は $\mathrm{O}(n+m)$ である．

3.1 一般解法による可逆化

Landauer 法で力任せ法を可逆化する．C 言語から Janus への変換規則を定めて変換することで可逆化する．変数宣言は C 言語で変数の初期値を定義していない場合，Janus では初期値を 0 とする．local で初期値が 0，型が int の変数 x を宣言し，プログラムの最後に x をスタックに push して値を 0 に初期化し，delocal で変数を解放する．代入は x をスタックに push して x を 0 に初期化し，x と e の排他的論理和を取ることで x の値を e にする．if 文は 1 番目の副文と 2 番目の副文を実行したときにそれぞれ 1，0 を push し，スタックの 1 番上の値を確認することで s_1, s_2 のどちらを実行したかを明らかにできる．for 文は i の値を 0 にし，s を実行して i を 1 増やす．$i < n$ なら 0，そうでなければ 1 を push し until top(g)=1 でスタックの 1 番上を確認する．1 番上が 1 ならば from 文を終わり，1 でなければ from 文の内容を繰り返す．while 文は 1 を push することで from 文の最初の式を top(g)=1 とすることで真にする．s を実行し，条件式 e が真なら 0，偽なら 1 を push し until top(g)=1 でスタックの 1 番上を確認する．1 番上が 1 ならば from 文を終わり，0 なら from 文の内容を繰り返す．次に，Bennett 法で力任せ法を可逆化する．Landauer 法で可逆化した力任せ法を実行後，逆実行してゴミを消去すると出力が 2 回行われる．値を出力せずに i の値を保存するスタックを用意し，実行後にスタックの中身を出力することで出力を 1 回にできる．

3.2 提案手法による可逆化

力任せ法を可逆化する際の問題点は t の $i+j$ 文字目と p の j 文字目が一致するかの文字列照合が失敗したとき，j の値を 0 に初期化する方法である．j を 0 に初期化する方法は 2 つ考えられる．1 つ目は照合が失敗した場合は $j = 0$ になるまで j の値を 1 ずつ減らす方法である．この方法を提案手法 1 とする．実装の際はパターンがテキストの位置 i に存在するかを判定するプロシージャを作成し，作成したプロシージャを逆実行することによって j の値を 1 ずつ減らす処理を行う．2 つ目は照合が失敗した場合も照合を $t_{i+m-1} = p_{m-1}$ まで続ける方法である．この方法を提案手法 2 とする．最後まで照合を続けた場合，j の値は m で確定するので，j と m の排他的論理和を計算し，j の値を 0 にすることができる．照合失敗と成功の区別をつけることができるようにテキストの位置 i にパターンが存在するかを判定するとき，テキストとパターンで不一致な文字を発見したらテキストの位置 i の値が負になるように値を引く．テキストの位置 i の値が負になるように引くことで照合終了時，テキストの位置 i の値が 0 以上ならばテキストの位置 i にパターンは存在し，負ならば存在しないと判定できる．パターンの最初の方で文字列照合が失敗した場合は提案手法 1 の方が少ない時間計算量で j を初期化することができ，パターンの最後の方で文字列照合が失敗した場合は提案手法 2 の方が時間計算量が少なくなると考える．各文字の出現する確率が等しいとすると文字列の最初の方で照合が失敗する確率が高いので，提案手法 1 の方が時間計算量が少なくなる場合が多いと考える．

4 Rabin–Karp 法

Rabin–Karp 法は文字列を整数に変換するハッシュ関数 h を用いて文字列照合を行う．Rabin–Karp 法はパターン p，テキスト t，それらの長さ m と n，アルファベットの種類の個数 b，ハッシュ関数 h で用いる整数 q が入力，パターンがテキストに存在する位置の列が出力である．パターン p のハッシュ値は，

$$h(p) = (p_0 b^{m-1} + p_1 b^{m-2} + \cdots + p_{m-1}) \bmod q \quad (1)$$

とする．t の i 文字目から連続する m 文字を抜き出した文字列のハッシュ値は，

$$h(t_it_{i+1}\ldots t_{i+m-1}) = (t_ib^{m-1} + t_{i+1}b^{m-2} + \cdots + t_{i+m-1}) \bmod q \quad (2)$$

とする．2 つの文字列のハッシュ値が等しい場合のみ力任せ法を用いて文字列照合をすることで p が t のどこに存在するかを調べることができる．

任意の i $(0 \le i < n-m)$ に対して，$h(t_it_{i+1}\ldots t_{i+m-1})$ から $h(t_{i+1}t_{i+2}\ldots t_{i+m})$ は定数時間 O(1) で求められる．$h_i = h(t_it_{i+1}\ldots t_{i+m-1})$ とする．h_i と i から $h(t_{i+1}t_{i+2}\ldots t_{i+m})$ への関数

$$f(h_i, i) = (b(h_i - t_ib^{m-1}) + t_{i+m}) \bmod q \quad (3)$$

を考える．b と q が互いに素ならば関数 f は第 1 引数について単射なので可逆な Rabin–Karp 法で利用できる．

補題 1 b と q が互いに素であるとき次が成り立つ：

$$f(h, i) = f(h', i) \Rightarrow h = h' \quad (4)$$

証明 f の定義より，

$$\begin{aligned} & f(h,i) = f(h',i) \\ \Leftrightarrow\ & b(h - t_ib^{m-1}) + t_{i+m} \\ & \equiv b(h' - t_ib^{m-1}) + t_{i+m} \pmod q \end{aligned}$$

である．両辺を展開し，$-t_ib^m + t_{i+m}$ を引くと

$$\Leftrightarrow\ bh \equiv bh' \pmod q$$

となる．両辺を b で割ると b と q は互いに素なので

$$\Leftrightarrow\ h \equiv h' \pmod q$$

が成り立つ．h，h' は q で割った余りなので $0 \le h, h' \le q-1$ である．よって，

$$h \equiv h' \pmod q \Leftrightarrow h = h'$$

である．したがって，$f(h,i) = f(h',i) \Rightarrow h = h'$ である．以上のことから，$f(h,i)$ は第 1 引数について単射である．□

Rabin–Karp 法は b^{m-1}, $h(p)$, $h(t_0t_1\ldots t_{m-1})$ を求める前処理に時間計算量 O(m) が必要である．文字列の比較にかかる最悪時間計算量は力任せ法と同じ O($m(n-m+1)$) となる．しかし，q の値を大きな素数とすると文字列の比較にかかる時間計算量は O($n+m$) である．

Rabin–Karp 法を可逆化する際の問題点は $h(p)$ の値と $h(t_it_{i+1}\ldots t_{i+m-1})$ $(0 \le i \le n-m)$ の値が等しくなり，2 つの文字列が一致しているかを確かめた結果不一致であることが判明したときである．なぜなら，力任せ法と同様に j をどのように初期化するかという問題が発生するからである．しかし，j の初期化は力任せ法と同様の方法で行うことができる．

Rabin–Karp 法を実装するとき，ハッシュ値を q で割った余りに更新できるようにハッシュ値を $hash$ として $hash$ を q で割った余りが初期値の変数 tmp を宣言し，$hash$ と tmp の値を入れ替えることで $hash$ の値を q で割った余りに更新し，tmp を解放することでハッシュ値を q で割った余りに更新できる．$h(p)$ を求めるステップのとき tmp を解放する方法を考える．$hashp$ を q で割った余りに更新する前のパターンのハッシュ値，tmp を $hashp$ を q で割った余りとする．$hashp$ と tmp の値を交換するので tmp の値は q で割った余りに更新する前のパターンのハッシュ値である．よって，$-p_0b^{m-1} - p_1b^{m-2} - p_2b^{m-3} - \ldots - p_{m-1}$ を tmp に足すことで tmp の値を 0 にし，tmp を解放できる．$h(t_0t_1\ldots t_{m-1})$ を求めるステップも $h(p)$ を求めるステップと同様にして tmp の値を 0 にできる．

次に，ハッシュ値を $h(t_it_{i+1}\ldots t_{i+m-1})$ から $h(t_{i+1}t_{i+2}\ldots t_{i+m})$ に更新するステップのとき，tmp をゼロクリアし解放する方法を考える．$h = t_ib^{m-1} + t_{i+1}b^{m-2} + t_{i+2}b^{m-3} + \ldots + t_{i+m-1}$，$hasht = b(h - t_ib^{m-1}) + t_{i+m}$，$tmp = hasht \bmod q$ とする．tmp と $hasht$ の値を交換するので h の値を求めることができれば tmp をゼロクリアすることができる．q を素数とし，b と q は互いに素として h を求める．tmp と $hasht$ の交換後，

$$hasht = b(h - t_ib^{m-1}) + t_{i+m} \bmod q \quad (5)$$

である．右辺を展開し，合同式を用いて表すと

$$\Leftrightarrow\ hasht \equiv bh - t_ib^m + t_{i+m} \pmod q \quad (6)$$

である．両辺から $-t_ib^m + t_{i+m}$ を引き，b^{q-2} をかけると

$$\Leftrightarrow\ b^{q-2}(hasht + t_ib^m - t_{i+m}) \equiv b^{q-2}bh \pmod q \quad (7)$$

である．フェルマーの小定理より

$$\Leftrightarrow\ b^{q-2}(hasht + t_ib^m - t_{i+m}) \equiv h \pmod q \quad (8)$$

である．h は 0 以上 $q-1$ 以下なので

$$h = b^{q-2}(hasht + t_ib^m - t_{i+m}) \bmod q \quad (9)$$

である．よって，事前に b^{q-2} の値を求めれば h の値を求めることができる．h の値を求めることができたのでハッシュ値を更新するステップで変数 tmp を解放することができる．$h(p)$ と $h(t_0t_1\ldots t_{m-1})$ を求めるステップとハッシュ値を $h(t_it_{i+1}\ldots t_{i+m-1})$ から $h(t_{i+1}t_{i+2}\ldots t_{i+m})$ に更新するステップの両方で変数 tmp を解放できた．

5 アルゴリズムの解析

3，4 章で可逆化したアルゴリズムの時間計算量，空間計算量，及びゴミの量を解析し解析結果の比較を行う．

5.1 漸近的な計算量による解析

可逆化したアルゴリズムの漸近的な時間計算量，空間計算量，及びゴミの量を解析しオーダー記法を用いて表す．

表 1　文字列照合法の解析結果

名前	時間計算量	空間計算量	ゴミの量
非可逆な力任せ法	$O(m(n-m+1))$	$O(n+m)$	なし
非可逆な Rabin–Karp 法	前処理 $O(m)$ 比較 $O(n+m)$	$O(n+m)$	なし
Landauer 法	$O(m(n-m+1))$	$O(m(n-m+1))$	$O(m(n-m+1))$
Bennett 法	$O(m(n-m+1))$	$O(m(n-m+1))$	$O(n+m)$ (入力のみ)
提案手法 1	$O(m(n-m+1))$	$O(n+m)$	$O(n+m)$ (入力のみ)
提案手法 2	$O(m(n-m+1))$	$O(n+m)$	$O(n+m)$ (入力のみ)
Rabin–Karp 法	前処理 $O(m+q)$ 比較 $O(n+m)$	$O(n+m)$	$O(n+m)$ (入力のみ)
Rabin–Karp 法 (繰り返し二乗法)	前処理 $O(m+\log q)$ 比較 $O(n+m)$	$O(n+m+\log q)$	$O(n+m)$ (入力のみ)

また，非可逆なアルゴリズムと可逆化したアルゴリズムの解析結果を表にまとめると表 1 になる．

表 1 をもとに非可逆なアルゴリズムの解析結果と比較を行うことで可逆化したアルゴリズムが faithful かを判定する．提案手法と非可逆な力任せ法を比較すると時間計算量，空間計算量，及びゴミの量について条件を満たしているので faithful である．Rabin–Karp の提案手法と非可逆な Rabin–Karp 法を比較すると前処理の時間計算量と空間計算量が条件を満たさないので繰り返し二乗法を用いた Rabin–Karp の提案手法は faithful ではなく，繰り返し二乗法を用いない場合も前処理の時間計算量が条件を満たさないので faithful ではない．

5.2　ステップ数と最大メモリ使用量の解析

漸近的な計算量を用いた解析では 2 つの提案手法の間に差がないので，ステップ数，最大メモリ使用量を用いて 2 つの提案手法の解析を行う．ステップ数を用いた解析では何が 1 ステップとなるかを決め，決めたステップが行われた回数を数える．最大メモリ使用量を用いた解析では何がどれだけメモリを使用するのかを決めることによってメモリが最大でどれだけ使われるかを求める．ステップ数を求めるとき，変数の宣言，解放，計算，条件式，アサーションの判定，値の出力を 1 ステップとする．計算は 1 度足し算や引き算などを行うたびに 1 ステップとする．最大メモリ使用量は変数を 1 つ使用する度にメモリを 1，長さ n の配列を使用した場合はメモリを n 使用したとする．何文字目で文字列照合が失敗したかによってステップの実行回数が変化する場合は文字列照合に成功したときの実行回数と 1 文字目で失敗したときの実行回数から m 文字目で失敗したときの実行回数までをそれぞれ求め，平均で何回実行されるかを求める．

提案手法 1 の平均ステップ数は $8n-8m+(5m^2n-5m^3+15mn-10m^2+13m)/2(m+1)+17$ であり，最大メモリ使用量は長さ $n+m+5$ である．提案手法 2 の平均ステップ数は S をテキストの中に存在するパターンの数とすると $-13m^2/2+13mn/2-3m/2+8n+S+11$ である．最大メモリ使用量は変数 $n+m+4$ である．

提案手法 1 と提案手法 2 の平均ステップ数を比較する．提案手法 1 と提案手法 2 の平均ステップ数から $8n$ を引き，両辺に $2(m+1)$ をかけ，2 つの式の係数を比較すると m^3 の係数は提案手法 2 の式の方が小さく m^2n の係数は提案手法 1 の方が小さい．よって，テキストとパターンの長さの差が大きいと提案手法 1 の方が平均ステップ数が小さくなる．反対に差が小さいと提案手法 2 の方が平均ステップ数が小さくなると考える．ただし，テキストの長さが短い場合は m^2n の影響が小さくなるのでテキストとパターンの長さの差が大きいが提案手法 2 の方が平均ステップ数が小さくなる場合があると考える．次に，提案手法 1 と提案手法 2 のメモリの最大使用量を比較すると提案手法 2 の方が 1 単位のメモリ分優れている．

6　おわりに

本研究は文字列照合法のうち力任せ法と Rabin–Karp 法について可逆化をする方法の提案と実装を行った．非可逆なアルゴリズムと可逆化したアルゴリズムの時間，空間計算量とゴミの量を解析し非可逆なアルゴリズムと可逆なアルゴリズムの比較，提案手法同士の比較を行った．

力任せ法は非可逆な力任せ法と比較して時間計算量及び空間計算量が変わらずゴミが入力のみとなるアルゴリズムを提案と実装ができた．Rabin–Karp 法はハッシュ値を更新する関数の単射性を証明し，空間計算量と前処理の時間計算量が悪化するが文字列照合を行うステップの時間計算量は非可逆な Rabin–Karp 法と同じになりゴミが入力のみとなるアルゴリズムの提案と実装ができた．今後は faithful な可逆な Rabin–Karp 法の提案と実装を行うことが課題である．本研究で提案した手法が他のアルゴリズムを可逆化するときにも応用できると考える．

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.), LNCS, Vol.9458, Springer, pp.407–426(2015).

[2] Bennett, C.H.: Logical Reversibility of Computation, *IBM J. Res. Dev.*, Vol.17, No.6, pp.525–532 (1973).

[3] Cormen, T., Leiserson, C., Rivest, R., et al.(著), 浅野 哲夫, 岩野 和生, 梅尾 博司ほか (訳): アルゴリズムイントロダクション, 近代科学社, 第 3 版総合版 (2013).

[4] Landauer, R.: Irreversibility and Heat Generation in the Computing Process, *IBM J. Res. Dev.*, Vol.5, No.3, pp.183–191(1961).

[5] 家崎 雄太, 水野 竣太郎: 可逆線形探索, 南山大学 2017 年度卒業論文 (2018).

[6] 森田 健一: 可逆計算, 近代科学社 (2012).