ヨセフスの問題とその逆問題に対する線形時間アルゴリズム

修士学位論文

ヨセフスの問題とその逆問題に対する線形時間アルゴリズム

東北大学 大学院情報科学研究科
システム情報科学専攻 篠原・吉仲研究室
博士課程前期 2 年の課程
石塚 将太

2022 年 2 月 8 日

目次

第 1 章

序論

1.1 はじめに

本論文では，ヨセフスの問題という数学パズルを扱う．古典的なヨセフスの問題は，次のように表現される．はじめに，n 人の兵士 $0, 1, \cdots, n-1$ が円状に並んでおり，兵士 0 だけが剣を持っている．剣を持っている兵士は，自分から数えて k 人先の兵士を処刑し，その次の兵士に剣を手渡す．処刑された兵士は，円から取り除かれる．この処刑プロセスは，全ての兵士が処刑されるまで繰り返される．ここで，古典的なヨセフスの問題とは，n と k が与えられた状況で，最後に処刑される兵士を答える問題のことである．例えば，$n = 7, k = 3$ の状況では，図 1.1 に示すように，$(3, 0, 5, 4, 6, 2, 1)$ の順番で処刑が行われる．したがって，この問題に対する答えは 1 である．古典的なヨセフスの問題については，漸化式を用いた $O(n)$ 時間のアルゴリズムが知られている [12].

古典的なヨセフスの問題は，これまで様々な変種が考えられてきた．Ruskey ら [10] は，兵士に体力 ℓ を持たせ ℓ 回斬られるまでは死亡しないという変種，**体力つきヨセフス**

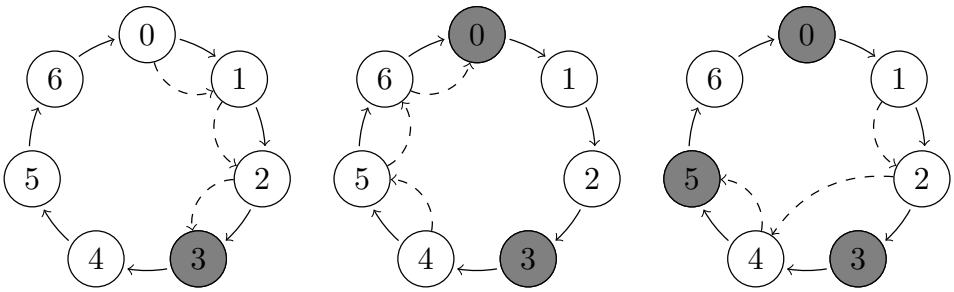

図 1.1　$n = 7, k = 3$ の場合の処刑の様子.

の問題 (The Feline Josephus Problem) を提案した．Matsumoto ら [7] は，兵士を円形に並べる代わりに直線上に並べ，一端に到達したら方向転換して数えるという変種を提案した．Sullivan ら [11] は，k 人おきに 1 人の兵士を処刑する代わりに，1 人おきに連続する k 人を処刑するという変種を提案した．

本論文では，古典的なヨセフスの問題において，最後に処刑される兵士ではなく，兵士が処刑される系列 $J_{n,k}$ に着目する．系列 $J_{n,k}$ のことを**ヨセフス順列**と呼び，ヨセフス順列を求める問題のことを**ヨセフスの問題**と呼ぶことにする．

Dowdy ら [3] は，任意の n に対して，$\mathrm{lcm}(1,2,\cdots,n)$ 個の異なるヨセフス順列が存在し，$J_{n,k} = J_{n,k+\mathrm{lcm}(1,2,\cdots,n)}$ を満たすことを示した．ここで，$\mathrm{lcm}(1,2,\cdots,n)$ は $\{1,2,\cdots,n\}$ の最小公倍数である．Knuth [4] は，ヨセフスの問題に対する $O(n\log n)$ 時間のアルゴリズムを 2 つ提案した．1 つは，平衡二分探索木を構築し，次に処刑する兵士を $O(\log n)$ 時間で計算するアルゴリズムである．もう 1 つは，ヨセフス順列の転倒に着目した分割統治法のアルゴリズムである．Lloyd [5] は，ヨセフスの問題に対して，$k < n$ の場合に動作する $O(n\log k)$ 時間のアルゴリズムを提案した．$\lceil n/k \rceil$ 個の完全二分木を構築し，次に処刑する兵士を $O(\log k)$ 時間で計算するアルゴリズムである．

本論文では，ヨセフスの問題に対する $O(n)$ 時間のアルゴリズムを提案する．ただし，このアルゴリズムは，$O(n\log n)$ ビットのビット列に対する論理演算および算術演算を定数時間で処理できるという仮定のもとで動作する．また，**ヨセフスの逆問題**を新たに定義し，同一の仮定のもとで $O(n)$ 時間で動作するアルゴリズムを提案する．さらに，体力つきヨセフスの問題について，関連するいくつかの課題を考え，取り組んだ結果について述べる．

1.2 構成

本論文の構成は次のようになっている．第 2 章では，ヨセフス順列の定義や，提案手法で使用するデータ構造の導入を行う．第 3 章では，ヨセフスの問題に対する線形時間アルゴリズムを提案する．第 4 章では，ヨセフスの逆問題に対する線形時間アルゴリズムを提案する．第 5 章では，体力つきヨセフスの問題に関連するいくつかの課題について述べる．第 6 章では，まとめと今後の課題について述べる．

第 2 章

準備

定義 1 (モジュラ逆数)**.** 整数 a, b が互いに素であるとき，$ax \equiv 1 \pmod{b}$ を満たす整数 $x \in \{0, 1, \cdots, b-1\}$ を，b **を法とした** a **のモジュラ逆数**といい，$(a)_b^{-1}$ と表記する．

命題 1. 整数 a, b が互いに素であるとき，$(a)_b^{-1}$ は $O(\log \min(a, b))$ 時間で計算可能である．

証明. 拡張ユークリッドの互除法 [2] により，整数 a, b が互いに素であるとき，$ax + by = 1$ を満たす整数 x, y は $O(\log \min(a, b))$ 時間で計算できる．ここで，x は $ax \equiv 1 \pmod{b}$ を満たすので，$(a)_b^{-1} = x \bmod b$ は $O(\log \min(a, b))$ 時間で計算可能である． □

集合のワード符号化

集合のワード符号化 (Word-encoded sets) [6] とは，正整数の有限部分集合 S をビット列に符号化して管理する手法である．S の最大要素を m とすると，集合 S は $O(m \log m)$ ビットのビット列として符号化される．より具体的には，S の各要素を $\lceil \log_2(m+1) \rceil + 1$ ビットの二進数で表記し，大きい順に連結することで，S を符号化する．例えば，$S = \{1, 3, 6\}$ のとき，S のワード符号は，$(0110\ 0011\ 0001)_2$ である．

この手法では，集合に対する操作を，ビット列に対する論理演算および算術演算の組み合わせで表現する．ビット長 x のビット列に対する論理演算および算術演算の計算時間の上界を $T(x)$ とすると，集合 S に対する次の 4 つの操作を $T(m \log m)$ 時間で実行することができる．

1. $\mathrm{rank}(a, S)$: $|\{x \in S \mid x < a\}|$ を返す．
2. $\mathrm{unrank}(r, S)$: $\mathrm{rank}(a, S) = r$ を満たす $a \in S$ を返す．
3. $\mathrm{insert}(a, S)$: S を $S \cup \{a\}$ に更新する．
4. $\mathrm{delete}(a, S)$: S を $S \setminus \{a\}$ に更新する．

ただし，$a \in S$ および $r \in \{0, 1, \cdots, |S| - 1\}$ である．

それぞれの操作の実装方法について述べる．ただし，自然数 m を集合 S の要素になりうる最大の自然数，自然数 b を $\lceil \log_2(m+1) \rceil + 1$，ビット列 $\mathbf{a}$ を集合 S のワード符号とする．また，ビット列 $\mathbf{a}$ の右端から数えて i ビット目の値を $\mathbf{a}[i]$ と表し，ビット列 $\mathbf{a}[j-1]\mathbf{a}[j-2]\cdots\mathbf{a}[i]$ を $\mathbf{a}[i, j]$ と表記する．さらに，自然数 a を二進数表記したビット列を $(a)_2$ で表す．例えば，$a = 38$ のとき，$(a)_2 = 100110$ であり，$(a)_2[0, 3] = 110$ である．

まず，ビット列 $\mathbf{a}[0, j]$ は，$\mathbf{a}$ と $2^j - 1$ との AND 演算を計算することで得られるので，$T(m \log m)$ で計算可能である．また，ビット列 $\mathbf{a}[i, j]$ は，$\mathbf{a}$ を右に i 回シフトしたビット列を $\mathbf{b}$ として，$\mathbf{b}[0, j-i]$ と表されるので，同じく $T(m \log m)$ 時間で計算可能である．

$\mathrm{unrank}(r, S)$ は，集合 S の要素の中で r 番目に小さい要素 a を返す操作である．要素 a はビット列 $\mathbf{a}$ の右端から数えて，rb ビット目から $(r+1)b-1$ ビット目に格納されているので，$\mathbf{a}[rb, (r+1)b]$ を計算することで得られる．

$\mathrm{insert}(a, S)$ は，集合 S に要素 a を追加する操作である．この操作は，次のようにして $T(m \log m)$ 時間で $\mathrm{rank}(a, S)$ に帰着される．$r = \mathrm{rank}(a, S)$ とすると，ビット列 $\mathbf{a}$ の右端から数えて，rb ビット目の直前に，$(a)_2$ を追加すればよい．ゆえに，自然数 a を追加した後のワード符号は，次の 3 つのビット列の算術和として得られる．

1. $\mathbf{a}[0, rb]$
2. $\mathbf{a} - \mathbf{a}[0, rb]$ を左に b 回シフトしたビット列
3. $(a)_2$ を右に rb 回シフトしたビット列

$\mathrm{delete}(a, S)$ は，集合 S から要素 a を削除する操作である．この操作は，次のようにして $T(m \log m)$ 時間で $\mathrm{rank}(a, S)$ に帰着される．$r = \mathrm{rank}(a, S)$ とすると，要素 a は $\mathbf{a}[rb, (r+1)b]$ に格納されている．ゆえに，要素 a を削除した後のワード符号は，次の 2 つのビット列の算術和として得られる．

1. $\mathbf{a}[0, rb]$
2. $\mathbf{a}$ を右に $(r+1)b$ 回シフトした後で左に rb 回シフトしたビット列

$\mathrm{rank}(a, S)$ は，a 未満である S の要素の個数を返す操作である．まず，$x \leq a < 2^b \Leftrightarrow 2^b \leq 2^b + a - x < 2^{b+1}$ であるから，$a < 2^b$ の条件下で，自然数 x が a 以下であることと，$(2^b + a - x)_2[b] = 1$ であることは同値である．そこで，$(2^b + a)_2$ を $|S|$ 回連結したビット列 $\mathbf{c}$ を考える．このとき，自然数 r に対して $a \geq \mathrm{unrank}(r, S)$ であることと，$(\mathbf{c} - \mathbf{a})[(r+1)b] = 1$ であることは同値である．したがって，a 以下である S の要素の個数は $\sum_{r=0}^{|S|-1} (\mathbf{c} - \mathbf{a})[(r+1)b]$ と計算できる．以上の議論において，a を $a-1$ に置き換えることで，$\mathrm{rank}(a, S)$ が得られる．

最後に，上述の $\mathrm{rank}(a, S)$ の計算が $T(m \log m)$ 時間で実行できることを示す．まず，ビット列 $\mathbf{c}$ は，$((2^b + a) \sum_{i=0}^{|S|-1} 2^{ib})_2$ と表すことができる．次に，$\mathbf{c} - \mathbf{a}$ と $(\sum_{i=0}^{|S|-1} 2^{ib})_2$ との AND 演算を計算すると，$\sum_{r=0}^{|S|-1} (\mathbf{c} - \mathbf{a})[(r+1)b] \cdot 2^{(r+1)b}$ が得られる．これを $2^b - 1$ で割った余りとして $\sum_{r=0}^{|S|-1} (\mathbf{c} - \mathbf{a})[(r+1)b]$ が得られる．ここで，$\sum_{i=0}^{|S|-1} 2^{ib} = \dfrac{2^{|S|b} - 1}{2^b - 1}$ であるから，以上の演算は全て $T(m \log m)$ 時間で実行できる．

本論文では，$\{0, 1, \cdots, n-1\}$ の部分集合を管理するために，この手法を利用する[*1]．このとき，符号の大きさは $O(n \log n)$ ビットであり，これはヨセフスの問題の出力サイズやヨセフスの逆問題の入力サイズに等しい．本論文全体を通して，$T(n \log n) = O(1)$ を仮定する．この仮定から，$\{0, 1, \cdots, n-1\}$ の部分集合に対して，上記の 4 つの操作は定数時間で実行可能である．

ヨセフス順列・ヨセフスの問題・ヨセフスの逆問題

兵士 $0, 1, \cdots, n-1$ が円形に並んでいる．兵士 0 から始めて，全ての兵士が処刑されるまで，k 人おきに兵士を処刑していく．処刑された兵士は円から取り除く．i 番目に処刑される兵士を s_i として，$J_{n,k} = (s_0, s_1, \cdots, s_{n-1})$ のことを**ヨセフス順列**と定義する．例

[*1] 実際は，$\{1, 2, \cdots, n\}$ の部分集合を管理し，全ての $i \in \{1, 2, \cdots, n\}$ に対して i を $i-1$ とみなして扱う．

えば，$n = 7, k = 3$ のとき，$J_{n,k} = (3, 0, 5, 4, 6, 2, 1)$ である（図 1.1 を参照）．**ヨセフスの問題**とは，自然数 n, k から $J_{n,k}$ を求める問題のことである．**ヨセフスの逆問題**とは，$\{0, 1, \cdots, n-1\}$ の順列 π から $J_{n,k} = \pi$ を満たす自然数 k を求める問題のことである．

第 3 章

ヨセフスの問題

本章では，ヨセフスの問題に対する線形時間アルゴリズムを提案する．

はじめに，$J_{n,k} = (s_0, s_1, \cdots, s_{n-1})$ を用いて，ランク列 $R_{n,k}$ を定義する．兵士 s_{i-1} の処刑後，まだ処刑されていない兵士の集合 S_i は $S_i = \{0, 1, \cdots, n-1\} \setminus \{s_0, s_1, \cdots, s_{i-1}\}$ と表される．ここで，S_i に属する兵士に対して，昇順で 0 から $n-i-1$ の番号を割り当てることを考える．特に，次に処刑される兵士 s_i は番号 $r_i = \mathrm{rank}(s_i, S_i)$ が割り当てられる．ここで，ランク列 $R_{n,k} = (r_0, r_1, \cdots, r_{n-1})$ を以下のように定義する．

$$\begin{cases} r_0 := s_0 \\ r_i := \mathrm{rank}(s_i, \bigcup_{j=0}^{n-1} \{j\} \setminus \bigcup_{j=0}^{i-1} \{s_j\}) \quad (i = 1, 2, \cdots, n-1) \end{cases} \tag{3.1}$$

例えば，$n = 7, k = 3$ のとき，$R_{n,k} = (3, 0, 3, 2, 2, 1, 0)$ である（図 3.1 を参照）．

次に，r_{i-1} と r_i の関係について考える．兵士 s_{i-1} は，処刑される直前において，r_{i-1} と番号付けされている．したがって，その 1 人先の兵士 t は，s_{i-1} の処刑の直後には，$r_{i-1} \bmod (n-i)$ と番号付けされることになる．兵士 s_i は兵士 t から数えて k 人先の兵士であるから，$r_i = (r_{i-1} + k) \bmod (n-i)$ が成立する．以上のことから，次の漸化式が導かれる．

$$\begin{cases} r_0 = k \bmod n \\ r_i = (r_{i-1} + k) \bmod (n-i) \quad (i = 1, 2, \cdots, n-1) \end{cases} \tag{3.2}$$

ここで，この漸化式を満たす数列 $(r_0, r_1, \cdots, r_{n-1})$ をランク列 $R_{n,k}$ であると再定義する．このとき，性質 (3.1) を利用して，次のようにヨセフス順列 $J_{n,k} = (s_0, s_1, \cdots, s_{n-1})$

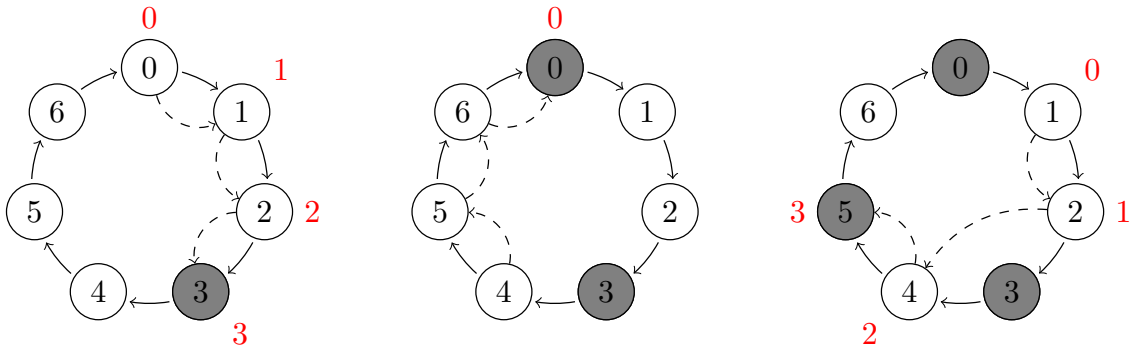

図 3.1　ランク列 $R_{7,3}$ 導出の様子．$J_{7,3} = (s_0, s_1, \cdots, s_6)$ とすると，左から順に兵士 $s_0 = 3, s_1 = 0, s_2 = 5$ が処刑される瞬間の様子を表している．また，兵士 s_i が処刑される直前において生き残っている兵士の集合 S_i に対して，昇順で 0 から $n - i$ の番号を割り振り，その一部を赤字で示している．

を形式的に定義しなおすことができる．

$$\begin{cases} s_0 := r_0 \\ s_i := \mathrm{unrank}(r_i, \bigcup_{j=0}^{n-1}\{j\} \setminus \bigcup_{j=0}^{i-1}\{s_j\}) \quad (i = 1, 2, \cdots, n-1) \end{cases} \tag{3.3}$$

最後に，ヨセフス順列 $J_{n,k}$ を高速に計算する手法について述べる．

定理 1. ヨセフスの問題は Algorithm 1 によって，$O(n)$ 時間で解くことができる．

証明. $J_{n,k} = (s_0, s_1, \cdots, s_{n-1})$ かつ $R_{n,k} = (r_0, r_1, \cdots, r_{n-1})$ とする．次の 3 つの手順によって，(r_i, S_i, s_i) から $(r_{i+1}, S_{i+1}, s_{i+1})$ を計算できる．

1. 漸化式 (3.2) を用いて，r_i から r_{i+1} を生成する．
2. 集合 S_i から要素 s_i を取り除くことで，集合 S_{i+1} を得る．
3. 式 (3.3) を用いて，r_{i+1} と S_{i+1} から s_{i+1} を得る．

集合 S_i をワード符号化して管理することで，上記の計算は定数時間で実行可能である．

ここで，$r_0 = k \bmod n$，$s_0 = k \bmod n$，$S_0 = \{0, 1, \cdots, n-1\}$ とすると，上記の計算を繰り返すことで，$J_{n,k} = (s_0, s_1, \cdots, s_{n-1})$ を $O(n)$ 時間で計算できる．　□

Algorithm 1 ヨセフス問題に対する線形時間アルゴリズム

Require: 自然数 n, k

Ensure: ヨセフス順列 $J_{n,k}$

1: $s_0 := k \bmod n$
2: $r_0 := k \bmod n$
3: $S := \emptyset$
4: **for** $i = 0$ to $n - 1$ **do**
5: insert(i, S)
6: **end for**
7: **for** $i = 1$ to $n - 1$ **do**
8: $r_i := (r_{i-1} + k) \bmod (n - i)$
9: delete(s_{i-1}, S)
10: $s_i :=$ unrank(r_i, S)
11: **end for**
12: **return** $(s_0, s_1, \cdots, s_{n-1})$

第 4 章

ヨセフスの逆問題

4.1 特殊な連立一次合同式

本節では，ヨセフスの逆問題を解く際に現れる特殊な連立一次合同式を，高速に解くための方法について述べる（補題 3）．この補題を証明するために，中国剰余定理を利用する．

命題 2 (中国剰余定理 [8])**.** $a_1, a_2, \cdots, a_n$ を整数，$b_1, b_2, \cdots, b_n$ を対ごとに素な正整数とする．連立一次合同式

$$x \equiv a_i \pmod{b_i} \quad (i = 1, 2, \cdots, n)$$

はただ一つの解

$$x \equiv \sum_{i=1}^{n} a_i \frac{M}{b_i} \left(\frac{M}{b_i} \right)_{b_i}^{-1} \pmod{M}$$

を持つ．ただし，$M := \prod_{i=1}^{n} b_i$ である．

次に，$b_1, b_2, \cdots, b_n$ が必ずしも対ごとに素でない状況を考える．

補題 1. $a_1, a_2, \cdots, a_n$ を整数，$b_1, b_2, \cdots, b_n$ を正整数とする．連立一次合同式

$$x \equiv a_i \pmod{b_i} \quad (i = 1, 2, \cdots, n)$$

は $L := \mathrm{lcm}(b_1, b_2, \cdots, b_n)$ を法として高々一つの解を持つ．

証明. 2 つの解 x_0, x_1 を持つとすると，

$$x_0 \equiv a_i \equiv x_1 \pmod{b_i} \quad (i = 1, 2, \cdots, n)$$

が成り立つ．これは，$x_0 - x_1$ が L の倍数であることを意味している．すなわち，

$$x_0 \equiv x_1 \pmod{L}.$$

である． □

次に，特に $(b_1, b_2, \cdots, b_n) = (1, 2, \cdots, n)$ である状況を考える．

補題 2. $a_1, a_2, \cdots, a_n$ を整数とする．連立一次合同式

$$x \equiv a_i \pmod{i} \quad (i = 1, 2, \cdots, n) \tag{4.1}$$

は高々一つの解

$$x \equiv \sum_{p \in P} a_p \frac{L}{p} \left(\frac{L}{p} \right)_p^{-1} \pmod{L} \tag{4.2}$$

を持つ．ただし，$L := \mathrm{lcm}(1, 2, \cdots, n) = \prod_{i=1}^{m} {p_i}^{e_i}$ かつ $P = \{{p_1}^{e_1}, {p_2}^{e_2}, \cdots, {p_m}^{e_m}\}$ であり，$p_1, p_2, \cdots, p_m$ は n 以下の異なる素数である．

証明. 式 (4.1) から一部分を抜き出した連立一次合同式

$$x \equiv a_{{p_i}^{e_i}} \pmod{{p_i}^{e_i}} \quad (i = 1, 2, \cdots, m)$$

を考える．命題 2 より，この連立一次合同式はただ一つの解 (4.2) を持つ．さらに，補題 1 より，連立一次合同式 (4.1) が解を持つとすると，その解は必ず解 (4.2) に一致する． □

最後に，解 (4.2) が $O(n)$ 時間で計算できることを示す．

補題 3. $a_1, a_2, \cdots, a_n$ を整数とする．連立一次合同式 (4.1) は，Algorithm 2 により，$O(n)$ 時間で解くことができる．

証明. 解 (4.2) の右辺を $O(n)$ 時間で計算できることを示せばよい．

まず，集合 P は次のようにして計算できる．

1. n 以下の素数の集合 $\{p_1, p_2, \cdots, p_m\}$ を計算する．これは $O(n/\log\log n)$ 時間で計算できることが知られている [1].
2. p_1 の冪乗を n 以下の範囲で列挙する．それらの最大値が ${p_1}^{e_1}$ である．

3. 同様にして，${p_2}^{e_2}, {p_3}^{e_3}, \cdots, {p_m}^{e_m}$ を得る．列挙される数は相異なる n 以下の自然数であるから，$O(n)$ 時間でこの処理を終えることができる．

また，命題 1 から，任意の $p \in P$ に対して，$\left(\frac{L}{p}\right)_p^{-1}$ は $O(\log n)$ 時間で計算可能である．

以上のことから，解 (4.2) は $O(n + m \log n)$ 時間で計算できる．ここで，$n > 1$ のとき，n 以下の素数の個数 m に対して，$m < 1.25506 \frac{n}{\log n}$ が成り立つことが知られている [9]．ゆえに，$O(n)$ 時間で解 (4.2) を計算することができる． □

4.2 アルゴリズム

本節では，ヨセフスの逆問題を $O(n)$ 時間で解くアルゴリズムについて述べる．

まず，ヨセフス順列 $J_{n,k}$ からランク列 $R_{n,k}$ を計算することから始める．

補題 4. ヨセフス順列 $(s_0, s_1, \cdots, s_{n-1})$ が与えられたとき，ランク列 $R_{n,k}$ を $O(n)$ 時間で計算できる．ここで，k は $J_{n,k} = (s_0, s_1, \cdots, s_{n-1})$ を満たす．

証明. 集合 S をワード符号化して管理し，$\{0, 1, \cdots, n-1\}$ で初期化する．集合 S に対する 2 つの操作 delete および rank は，定数時間で行うことができるので，ランク列 $R_{n,k}$ は式 (3.1) にしたがって $O(n)$ 時間で計算できる． □

次に，ヨセフスの逆問題の解 k は，ある連立一次合同式の解であることを示す．

補題 5. $R_{n,k} = (r_0, r_1, \cdots, r_{n-1})$ とすると，次の式が成り立つ．

$$\begin{cases} k \equiv r_0 \pmod{n} \\ k \equiv r_i - r_{i-1} \pmod{n-i} \quad (i = 1, 2, \cdots, n-1) \end{cases} \tag{4.3}$$

証明. 漸化式 (3.2) を変形することで得られる． □

ここで，k は未知数であるから，式 (4.3) は k についての連立一次合同式であると捉えることができる．この連立一次合同式は補題 3 が適用できる形をしている．したがって，次の定理が成立する．

定理 2. ヨセフスの逆問題は $O(n)$ 時間で解くことができる．

Algorithm 2 連立一次合同式 (4.1) を解く線形時間アルゴリズム

Require: 正整数 n および整数 $a_1, a_2, \cdots, a_n$

Ensure: 連立一次合同式 $x \equiv a_i \pmod{i} \quad (i = 1, 2, \cdots, n)$ の解

1: $P :=$ the empty set
2: $L := 1$
3: Generate primes $p_1, p_2, \cdots, p_m$ less than or equal to n
4: **for** $i = 1$ to m **do**
5: $\quad p := p_i$
6: $\quad$ **while** $p \times p_i \leq n$ **do**
7: $\quad\quad p := p \times p_i$
8: $\quad$ **end while**
9: $\quad$ Insert p into P
10: $\quad L := L \times p$
11: **end for**
12: $x := 0$
13: **for** $p \in P$ **do**
14: $\quad x := x + a_p \times \frac{L}{p} \times \left(\frac{L}{p}\right)_p^{-1}$
15: **end for**
16: **if** $x \equiv a_i \pmod{i}$ for all $i \in \{1, 2, \cdots, n\}$ **then**
17: $\quad$ **return** x
18: **else**
19: $\quad$ **return** null
20: **end if**

証明. $\{0, 1, \cdots, n-1\}$ の順列 π が与えられたとき，順列 π をヨセフス順列とみなして，補題 4 の手順により，数列 R を得る．次に，数列 R をランク列とみなして，連立一次合同式 (4.3) を解く．このとき，次の 2 つを示すことができる．

I. 連立一次合同式 (4.3) の解 k が存在するならば，$\pi = J_{n,k}$ である．
II. 連立一次合同式 (4.3) の解が存在しないならば，$\pi = J_{n,k}$ を満たす k は存在し

ない．

（I の証明）数列 R が解 k に対して連立一次合同式 (4.3) を満たすことから，数列 R は式 (3.3) を満たす．すなわち，$R = R_{n,k}$ である．また，数列 R は補題 4 の手順にしたがって生成されているので，

$$\begin{cases} r_0 = \pi_0 \\ r_i = \mathrm{rank}(\pi_i, \bigcup_{j=0}^{n-1}\{j\} \setminus \bigcup_{j=0}^{i-1}\{\pi_j\}) \quad (i = 1, 2, \cdots, n-1) \end{cases}$$

を満たす．これを変形して，

$$\begin{cases} \pi_0 = r_0 \\ \pi_i = \mathrm{unrank}(r_i, \bigcup_{j=0}^{n-1}\{j\} \setminus \bigcup_{j=0}^{i-1}\{\pi_j\}) \quad (i = 1, 2, \cdots, n-1) \end{cases}$$

を得る．ヨセフス順列 $J_{n,k}$ の定義式 (3.3) と比較すると，$\pi = J_{n,k}$ である．

（II の証明）対偶を示す．すなわち，$\pi = J_{n,k}$ を満たす k が存在するならば，連立一次合同式 (4.3) は解を持つことを示す．$\pi = J_{n,k}$ を満たす k が存在するとき，補題 4 より，$R = R_{n,k}$ である．したがって，補題 5 より，連立一次合同式 (4.3) は必ず解を持つ．

以上のことから，補題 3 より，連立一次合同式 (4.3) を $O(n)$ 時間で解くことにより，ヨセフスの逆問題を解くことができる． □

第 5 章

体力つきヨセフスの問題

本章では，**体力つきヨセフスの問題** (The Feline Josephus Problem) [10] について述べる．体力つきヨセフスの問題では，兵士が処刑のプロセスにおいて，ℓ 回斬られるまで死亡しないという拡張を考える．ヨセフスの問題と同様に，i 番目に死亡する兵士を s_i として，順列 $J_{n,k,\ell} = (s_0, s_1, \cdots, s_{n-1})$ を定義する．また，i 番目に斬られる兵士を h_i として，数列 $H_{n,k,\ell} = (h_0, h_1, \cdots, h_{n\ell-1})$ を定義する．例えば，$n = 6, k = 3, \ell = 2$ のときの処刑の様子を図 5.1 に示した．$H_{6,3,2} = (3, 1, 5, 3, 1, 0, 0, 2, 4, 5, 5, 4, 2)$ であり，$J_{6,3,2} = (3, 1, 0, 5, 4, 2)$ である．

5.1 (n, k, ℓ) から $H_{n,k,\ell}$ および $J_{n,k,\ell}$ を計算するアルゴリズム

本節では，(n, k, ℓ) から $H_{n,k,\ell}$ および $J_{n,k,\ell}$ を計算する問題について述べる．Ruskey ら [10] は，この問題を解く $O(n^2)$ 時間アルゴリズムの存在を論文中で示している．本節

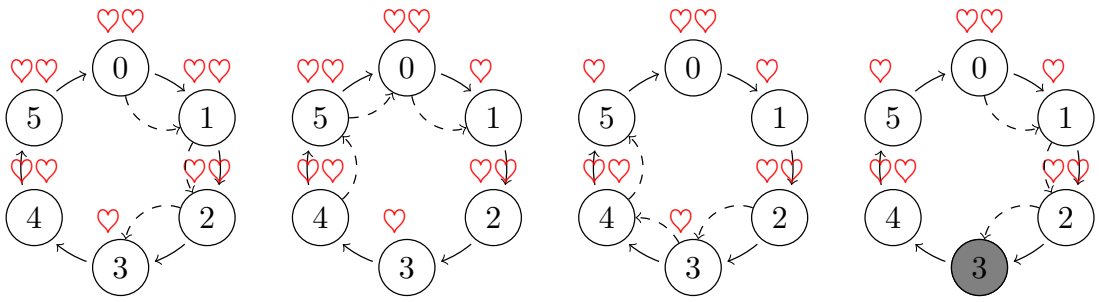

図 5.1　$H_{6,3,2}$ および $J_{6,3,2}$ 導出の様子．左から順に兵士 $3, 1, 5, 3$ が斬られた瞬間を表しており，♡ は兵士の残り体力を表す．

では，$l < n$ の条件下で Ruskey らのアルゴリズムよりも高速に動作する $O(n\ell)$ 時間のアルゴリズムを提案する．提案するアルゴリズムは，ヨセフスの問題を解く線形時間アルゴリズムを自然に拡張したアルゴリズムである．

定理 3. (n, k, ℓ) が与えられたとき，$H_{n,k,\ell}$ および $J_{n,k,\ell}$ を Algorithm 3 によって，$O(n\ell)$ 時間で計算することができる．

証明. 全ての要素が ℓ である長さ n の配列 A を用意する．要素 $A[i]$ は，現時点の兵士 i の残りの体力を管理している．また，集合 $\{0, 1, \cdots, n-1\}$ のワード符号を S とする．S は，現時点で生き残っている兵士の集合を管理する．

$H_{n,k,\ell}$ の要素を，逐次的に生成していく過程で，$J_{n,k,\ell}$ を計算する．斬られる兵士を x とすると，$A[x]$ から 1 を引くことで配列 A を正しく更新できる．要素 $A[x]$ を更新し，$A[x] = 0$ となった場合に，集合 S から要素 x を取り除くことで，S を正しく更新できる．集合 S から取り除いた要素を，取り除いた順番に並べることで，$J_{n,k,\ell}$ が得られる．

以下は，配列 A および集合 S が上記の手順によって，正しく更新されている前提のもとで考える．また，i 番目の処刑で h_i が切られる直前の配列 A を A_i，集合 S を S_i と表記する．兵士 h_i および $\mathrm{rank}(h_i, S_i)$ が既知であるときに，兵士 h_{i+1} および $\mathrm{rank}(h_{i+1}, S_{i+1})$ が定数時間で計算できることを示す．

i 番目の処刑で兵士 h_i が切られる直前において，h_i の 1 つ先の兵士を t とすると，$\mathrm{rank}(t, S_{i+1})$ の値は，$A_i[h_i]$ の値によって分岐する．$A_i[h_i] = 1$ のとき，h_i は斬られると死亡するので，$\mathrm{rank}(t, S_{i+1}) = \mathrm{rank}(h_i, S_i) \bmod (n-i)$ である．$A_i[h_i] > 1$ のとき，h_i は斬られても死亡しないので，$\mathrm{rank}(t, S_{i+1}) = (\mathrm{rank}(h_i, S_i) + 1) \bmod (n-i)$ である．いずれの場合も，兵士 t の k 人先が次に斬られる兵士 h_{i+1} であるから，$\mathrm{rank}(h_{i+1}, S_{i+1}) = (\mathrm{rank}(t, S_{i+1}) + k) \bmod (n-i)$ である．最後に，$h_{i+1} = \mathrm{unrank}(\mathrm{rank}(h_{i+1}, S_{i+1}), S_{i+1})$ より h_{i+1} が得られる．

$h_0 = \mathrm{rank}(h_0, S_0) = k \bmod n$ から始めて，上記の計算を繰り返すことで，$H_{n,k,\ell}$ および $J_{n,k,\ell}$ を $O(n\ell)$ 時間で計算できる． □

Algorithm 3 (n, k, ℓ) から $H_{n,k,\ell}$ および $J_{n,k,\ell}$ を計算する $O(n\ell)$ 時間アルゴリズム

Require: 正整数 n, k, ℓ

Ensure: 数列 $H_{n,k,\ell}$, 順列 $J_{n,k,\ell}$

$A := \underbrace{(\ell, \ell, \cdots, \ell)}_{n \text{ 個}}$
$J := ()$
$S := \emptyset$
for $i = 0$ to $n - 1$ **do**
    insert(i, S)
end for
$h_0 := k \bmod n$
$r_0 := k \bmod n$
for $i = 0$ to $n\ell - 1$ **do**
    $A[h_i] := A[h_i] - 1$
    if $A[h_i] = 0$ **then**
        $r_{i+1} := (r_i + k) \bmod (n - i)$
        delete(h_i, S)
        append h_i to J
    else
        $r_{i+1} := (r_i + k + 1) \bmod (n - i)$
    end if
    $h_{i+1} :=$ unrank(r_{i+1}, S)
end for
return $(h_0, h_1, \cdots, h_{n\ell-1}), J$

5.2 $J_{n,k,\ell}$ に関する数え上げ

本節では，$J_{n,k,\ell}$ に関する数え上げについて述べる．ヨセフスの問題では，任意の n に対して $|\{J_{n,k} \mid k \in \mathbb{N}\}| = \mathrm{lcm}(1, 2, \cdots, n)$ であることが知られている [3]．ただし，$\mathrm{lcm}(1, 2, \cdots, n)$ は $\{1, 2, \cdots, n\}$ の最小公倍数である．

表 5.1　$f(n) := \left|\{J_{n,k,\ell} \mid k \in \mathbb{N}, \ell \in \mathbb{Z}^+\}\right|$

n	1	2	3	4	5	6	7	8	9	10
	1	2	6	12	60	69	429	908	2738	3501

表 5.2　$g(n,k) := \left|\{J_{n,k,l} \mid l \in \mathbb{Z}^+\}\right|$

n \ k	1	2	3	4	5	6	7	8	9	10	11	12
1	1											
2	1	1										
3	1	1	1	1	1	1						
4	1	1	1	1	1	1	1	1	1	2	1	1

$J_{n,k,\ell}$ についても，同様の数え上げを行うことを考える．以下の 3 つの場合について考察したが，いずれの場合も閉じた式は得られなかった．

- $f(n) := |\{J_{n,k,\ell} \mid k \in \mathbb{N}, \ell \in \mathbb{Z}^+\}|$
- $g(n,k) := |\{J_{n,k,\ell} \mid \ell \in \mathbb{Z}^+\}|$
- $h(n,\ell) := |\{J_{n,k,\ell} \mid k \in \mathbb{N}\}|$

それぞれの場合について，n, k, ℓ を固定して，実験的に計算した値を表 5.1, 5.2, 5.3 にまとめる．

5.3　$J_{n,k,\ell}$ から (n,k,ℓ) を計算するアルゴリズム

本節では，体力つきヨセフスの問題の逆問題を考える．より正確には，順列 π が与えられたときに，$J_{n,k,\ell} = \pi$ を満たす (n,k,ℓ) を返す問題を考える．

表 5.3 の $\ell = 1$ の行と表 5.1 とを比べることで，ある順列 π について，$J_{n,k,\ell} = \pi$ であるが $J_{n,k} = \pi$ を満たす (n,k) が存在しないことが分かる．例えば，$\pi = (3,1,0,4,2,5)$ とすると，$J_{6,27,2} = \pi$ だが，$J_{n,k} = \pi$ を満たす (n,k) は存在しない．したがって，この逆問題は単にヨセフスの逆問題を解くだけでは答えることはできない．

具体的に，逆問題を解くアルゴリズムを与えることはできなかった．

表 5.3　$h(n,l) := |\{J_{n,k,l} \mid k \in \mathbb{N}\}|$

l \ n	1	2	3	4	5	6	7	8	9	10
1	1	2	6	12	60	60	420	840	2520	2520
2	1	2	6	11	59	50	410	662	2166	2183
3	1	2	6	11	59	48	408	633	2028	1999
4	1	2	6	11	59	48	408	622	1998	1941
5	1	2	6	11	59	48	408	622	1992	1934
6	1	2	6	11	59	48	408	622	1992	1934
7	1	2	6	11	59	48	408	622	1992	1934
8	1	2	6	11	59	48	408	622	1992	1934
9	1	2	6	11	59	48	408	622	1992	1934
10	1	2	6	11	59	48	408	622	1992	1934

第 6 章

まとめ

本論文の主な成果は以下の 2 点である．

1. ヨセフスの問題に対する $O(n)$ 時間のアルゴリズムを提案した．
2. ヨセフスの逆問題を提案し，$O(n)$ 時間のアルゴリズムを提案した．

ただし，$O(n \log n)$ ビットのビット列に対する論理演算および算術演算を定数時間で処理できるという強い仮定が必要である．より弱い仮定のもとで，$O(n)$ 時間のアルゴリズムを提案することが今後の課題である．

また，体力つきヨセフスの問題についても，処刑された兵士を順番に並べた順列 $J_{n,k,\ell}$ を考え，$J_{n,k,\ell}$ に関連する次の研究課題を設定したが，いずれも未解決である．

- (n, k, ℓ) から $J_{n,k,\ell}$ を計算する $O(n)$ 時間のアルゴリズムを提案すること．
- 次の種類数を数え上げること．
  - $f(n) := |\{J_{n,k,l} \mid k \in \mathbb{N}, l \in \mathbb{Z}^+\}|$
  - $g(n, k) := |\{J_{n,k,l} \mid l \in \mathbb{Z}^+\}|$
  - $h(n, \ell) := |\{J_{n,k,l} \mid k \in \mathbb{N}\}|$
- 順列 π から $J_{n,k,\ell} = \pi$ を満たす (n, k, ℓ) を返す問題に対する効率的なアルゴリズムを提案する．

参考文献

[1] A. O. L. Atkin and D. J. Bernstein. Prime sieves using binary quadratic forms. *Mathematics of Computation*, Vol. 73, No. 246, pp. 1023–1030, 2004.

[2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 2nd edition, 2001.

[3] James Dowdy and M. Mays. Josephus permutations. *JCMCC. The Journal of Combinatorial Mathematics and Combinatorial Computing*, Vol. 6, pp. 125–130, 1989.

[4] Donald E. Knuth. *The Art of Computer Programming. Volume III. Searching and Sorting*. The Art of Computer Programming. Addison-Wesley, 1973.

[5] Errol L Lloyd. An $O(n \log m)$ algorithm for the Josephus Problem. *Journal of Algorithms*, Vol. 4, No. 3, pp. 262–270, 1983.

[6] Martin Mareš and Milan Straka. Linear-time ranking of permutations. In *Proceedings of the 15th Annual European Conference on Algorithms*, ESA'07, pp. 187–193, Berlin, Heidelberg, 2007. Springer-Verlag.

[7] Keiichi Matsumoto, Tomoki Nakamigawa, and Mamoru Watanabe. On the switchback vertion of Josephus problem. Yokohama Mathematical Journal = 横濱市立大學紀要. D 部門, 数学, Vol. 53, No. 2, pp. 83–88, 2007.

[8] K.H. Rosen. *Elementary Number Theory and Its Applications*. Addison-Wesley mathematics series. Addison-Wesley Publishing Company, 1984.

[9] J. Barkley Rosser and Lowell Schoenfeld. Approximate formulas for some functions of prime numbers. *Illinois Journal of Mathematics*, Vol. 6, No. 1, pp. 64–94, 1962.

[10] Frank Ruskey and Aaron Williams. The feline Josephus problem. *Theory of Computing Systems*, Vol. 50, No. 1, pp. 20–34, 2012.

[11] Shaun Sullivan and Thomas Beatty. Structured shuffles and the Josephus problem. *Open Journal of Discrete Mathematics*, Vol. 2, No. 4, pp. 138–141, 2012.

[12] Diana Christine Woodhouse. The extended Josephus problem. In *Revista matemática hispanoamericana*, Vol. 33, pp. 207–218, 1973.

謝辞

本論文の執筆にあたり，多くのご指導を賜りました，東北大学大学院情報科学研究科 篠原歩教授，吉仲亮准教授，ならびに Diptarama Hendrian 助教に心より感謝申し上げます．先生方には，本研究のみならず，学士課程，博士前期課程の 3 年間に渡って，様々な場面で的確なご助言を賜りました．

また，本論文の副審査委員を務めていただきました，東北大学大学院情報科学研究科 静谷啓樹教授ならびに伊藤健洋教授には，貴重なご意見をを賜りましたことを厚く御礼申し上げます．

篠原・吉仲研究室の先輩・同期・後輩の皆様には，研究に関わらず多くの面で支えていただき，大変充実した 3 年間を送ることができました．本当にありがとうございました．

最後に，学生生活を支えてくださった家族に心から感謝申し上げます．