Every Two-times Sorting Sequence Falls onto a Cycle

Makoto Sakuta, Makoto Araya, Zahidul Haque and Hiroyuki Iida
Department of Computer Science, Shizuoka University
E-mail: {sakuta,araya,zahi,iida}@cs.inf.shizuoka.ac.jp

Abstract

A two-times sorting sequence is constructed by calculating the next number with doubling the current number and sorting its digits into non-decreasing order. We propose the appropriate representation of a number with the arbitrary length of digits and the algorithms to calculate the sequence. Using this representation, we investigate two important properties of the sequence, i.e., cyclic and inductive. By the examination of the computer program, it is proved that every two-times sorting sequence falls onto a periodic cycle with the cyclic length $1, 2, 3, 4, 5, 6$, or 12.
Keyword: integer sequence, two-times sorting sequence, cyclic sequence, inductive sequence

1 Introduction

The characteristics of various mathematical sequences have been studied. For instance, the $3x+1$-problem is one of the famous sequences. In this paper, we study what we call the *two-times sorting sequence*. This sequence is proposed by Bottomley in the Web page ("ATS: Add Then Sort" in sequence A057615 [4]) and by Kotani in GPCC2000 [3], independently.

Definition 1 A two-times sorting sequence is constructed by the following procedure.

1. Start from any natural number.
2. Double the number.
3. Sort the digits of the number in the decimal format with non-decreasing order. (The digit zeros are removed.) The obtained number is the next number of the sequence.
4. Go to 2.

If the digits of a number in the decimal format are sorted with non-decreasing order, the number is called a *sorted number*. Otherwise, the number is called a *non-sorted number*. Let n be a natural number and m be the sorted number after sorting the digits of n. Then we write $sort(n) := m$. In a two-times sorting sequence $\{a_n\}$, $a_i = sort(2 \times a_{i-1})$ for all $i \geq 1$.

The succession of distinct non-zero digits in a sorted number is called a *type* of the sorted number. A type of a sorted number m is denoted by $type(m)$, which is represented as one of the 511 combinations of 1 to 9, that is, 1, 2, ..., 9, 12, ..., 89, 123, ..., 789, ..., 123456789. ◇

Example 1 In a two-times sorting sequence $\{a_n\}$, let $a_i = 11115568889$. Then, $type(a_i) = 15689$. The next number a_{i+1} is calculated as follows.

$2 \times a_i = 22231137778$, $a_{i+1} = sort(2 \times a_i) = 11222337778$, $type(a_{i+1}) = 12378$. ◁

As a basis, a cyclic sequence is defined.

Definition 2 The number sequence $\{a_n\}$ is said to fall onto a *periodic cycle* or simply called *cyclic* if the equations

$$
\begin{array}{lllllll}
a_i & = & a_{i+\ell} & = & a_{i+2\ell} & = & \cdots, \\
a_{i+1} & = & a_{i+\ell+1} & = & a_{i+2\ell+1} & = & \cdots, \\
\cdots & & \cdots & & \cdots & & \cdots, \\
a_{i+\ell-1} & = & a_{i+2\ell-1} & = & a_{i+3\ell-1} & = & \cdots
\end{array}
$$

hold for some i and ℓ. The smallest ℓ is called the *cyclic length* of $\{a_n\}$, while the smallest i is called the *length before cyclic.* ◇

The purpose of this paper is to prove the following theorem.

Theorem 1 Every two-times sorting sequence falls onto a periodic cycle and its cyclic length is 1, 2, 3, 4, 5, 6 or 12.

In our preliminary study, the convergent two-times sorting sequence, of which cyclic length is 1, was determined without using a computer [1, 2]. In addition, we made a computer program to calculate the two-times sorting sequences up to a certain limit, i.e., 100,000,000 [2]. We confirmed that all sequences up to the above limit resulted in cycles and their cyclic lengths were $1, 2, 3, 4, 5, 6$ or 12. However, since the natural number has no upper bound, the approach with using such a program will never lead to the rigorous proof of the theorem.

Owing to the page restriction, some proofs and examples are omitted in this paper. For details, see the full paper to be published later.

2 Data Representation

In this section, we introduce the data representation of a sorted number in order to deal with a number with the arbitrary length of digits. Even if the initial number a_0 of a sequence is not sorted, all the following numbers of the sequence are sorted numbers. Hereafter, we may assume that any initial number a_0 is a sorted number.

2.1 Data representation of sorted numbers

Definition 3 Any sorted number m is decomposed into each *fixed part* $C_m(d)$ and each *stretch part* $D_m(d)$ for all $d \in \{1, 2, ..., 9\}$. A fixed part $C_m(d)$ indicates a fixed number of figures of the digit d, while a stretch part $D_m(d)$ indicates the arbitrary length of the digit d with the lower bound.

Each stretch part is represented by a *stretch variable* L_t. The lower bound of a stretch part $D_m(d)$ is denoted by $\mathrm{LB}(D_m(d))$. Similarly, the lower bound of a stretch variable L_t in a sorted number m is denoted by $\mathrm{LB}_m(L_t)$. That is, $D_m(d) = L_t \geq \mathrm{LB}(D_m(d)) = \mathrm{LB}_m(L_t)$. If the stretch part of the digit d does not exist, $D_m(d) = 0$.

$\ell_m(d)$ is defined as the number of figures of the digit d in m. Obviously, $\ell_m(d) = D_m(d) + C_m(d)$. ◇

Example 2 Let $a_0 = 1...12336$ be a sorted number of which the number of figures of the digit 1 is equal to or greater than 3. Then, a_0 is represented as $D_{a_0}(1) = L_1 \geq \mathrm{LB}(D_{a_0}(1)) = \mathrm{LB}_{a_0}(L_1) = 3$, $C_{a_0}(2) = 1$, $C_{a_0}(3) = 2$, $C_{a_0}(6) = 1$ and others are 0. With using the vector notation, a_0 is also represented as $(\ L_1\ \ 1\ \ 2\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ )$. The number $a_1 = sort(2 \times a_0) = 2...2467$ is represented as $D_{a_1}(2) = L_1$, $C_{a_1}(2) = 1$, $C_{a_1}(4) = C_{a_1}(6) = C_{a_1}(7) = 1$ and others are 0. Here, $\ell_{a_1}(2) = L_1 + 1$. By the vector notation, $a_1 = (\ 0\ \ L_1 + 1\ \ 0\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0\ \ 0\ )$. Note that L_1 is an arbitrary natural number that is equal to or greater than 3 but has no upper bound. ◁

Thus, we can deal with any sorted number with the arbitrary length. The calculation of a sequence is performed with using the above data representation.

Definition 4 The minimum value of fixed parts related to a stretch variable L_t in a sorted number m is defined as $\mathrm{minC}_m(L_t)$. That is, $\mathrm{minC}_m(L_t) = \min\{C_m(d) \mid D_m(d) = L_t,\ d = 1, ..., 9\}$.

If the stretch part of a digit is a stretch variable, the count of figures of the digit should be positive. That is, $\mathrm{LB}_m(L_t) + \mathrm{minC}_m(L_t) \geq 1$. Moreover, the minimum value of $\mathrm{minC}_{a_k}(L_t)$ $(i \leq k \leq j)$ from a_i to a_j in a sequence $\{a_n\}$ is defined as $\mathrm{minC}_{a_i}^{a_j}(L_t)$. ◇

2.2 Properties of sorted numbers

In this subsection, the basic properties of the representation of sorted numbers are given. First, we introduce a concept that represents a relation between a sorted number in the above format and every particular number in the possible set of the sorted number.

Definition 5 Let m be a sorted number and $\{d_1, ..., d_k\}$ be the digits having the non-zero stretch parts with the stretch variables $\{L_1, ..., L_u\}$. Suppose that $D_m(d_j) = L_{t_j}$ for $1 \leq j \leq k$.
We consider two following infinite sets:
$S = \{(L_1 \leftarrow c_1, ..., L_u \leftarrow c_u) \mid c_1 \geq \mathrm{LB}_m(L_1), ..., c_u \geq \mathrm{LB}_m(L_u)\}$, and
$M = \{m_p \mid \ell_{m_p}(d_j) = c_j,\ c_j \geq \mathrm{LB}_m(L_{t_j}) + C_m(d_j)$ for $1 \leq j \leq k$, and $\ell_{m_p}(d) = C_m(d)$ for other $d\}$.
Each element in the set M is called an *instance* of m. M is called a *set of possible instances* of m. Each element in the set S is called an *assignment* of the stretch variables in m. S is called a *set of possible assignments* of the stretch variables in m. Obviously, M and S are bijective. ◇

Example 3 Let m be a sorted number (2 0 0 L_1+3 0 0 L_2-2 0 0), $L_1 \geq 1$ $L_2 \geq 4$. Then 11444477, 11444444777 are instances of m, while 114447777, 11444447 are not. A set of possible instances of m is $\{11444477, 114444477, ..., 114444777, 1144444777, ..., 1144447777, 11444447777, ...\}$. The corresponding set of possible assignments is $\{(L_1 \leftarrow 1, L_2 \leftarrow 4), (L_1 \leftarrow 2, L_2 \leftarrow 4), ..., (L_1 \leftarrow 1, L_2 \leftarrow 5), (L_1 \leftarrow 2, L_2 \leftarrow 5), ..., (L_1 \leftarrow 1, L_2 \leftarrow 6), (L_1 \leftarrow 2, L_2 \leftarrow 6), ...\}$. ◁

Second, we introduce the concepts of generality and equivalence of a sorted number.

Definition 6 A sorted number m_1 is more *general* than a sorted number m_2 (or m_2 is more *specific* than m_1) if a set M_2 of possible instances of m_2 is a subset of a set M_1 of possible instances of m_1. In other words, m_1 *covers* a specific m_2. ◇

Definition 7 Any two sorted numbers m_1 and m_2 are said to be *equivalent* iff both sets of possible instances are the same. In addition, each digit of m_1 and m_2 is said to be equivalent. The equivalence relation is denoted by $m_1 \equiv m_2$. ◇

Lemma 1 Let m_1 and m_2 be two sorted numbers such that
for all $d \in \{1, 2, ..., 9\}$: $C_{m_1}(d) = C_{m_2}(d)$, $D_{m_1}(d) = D_{m_2}(d)$, $\mathrm{LB}(D_{m_1}(d)) = \mathrm{LB}(D_{m_2}(d))$.
Then, $m_1 \equiv m_2$. □

3 Algorithms for Enumerating the Sequence Trees

We have made a program that enumerates the sequences for all the 511 types of the initial numbers, i.e., the combinations of 1 to 9.

3.1 Calculation of the next number

Lemma 2 Let the pair $(d, d') \in P = \{(5,1), (1,2), (6,3), (2,4), (7,5), (3,6), (8,7), (4,8), (9,9)\}$ represent shifting of digits. Let m be an instance of a sorted number. Then the next number m' in a sequence is calculated as:
for all $d' \in \{1, ..., 9\}$: $\ell_{m'}(d') = \ell_m(d) + x(type(m), d')$.
Here $x(type(m), d')$ represents a displacement for the digit d' in m'. □

Remark 1 The displacements can be calculated by applying the following procedure for every type t. First, make a sorted number m that has two figures for each digit of the type t respectively. Then, make a new number m' only by shifting the digits of m according to $(d, d') \in P$. On the other hand, $m'' = sort(2 \times m)$ is calculated by the normal addition and sorting. A displacement of the digit d under the type t is $\ell_{m''}(d) - \ell_{m'}(d)$.
The results obtained by the above procedure show that every displacement is -1, 0, or 1.

Considering Lemma 2, a sophisticated method for calculating the next number is given with using the data representation described in Section 2.

Definition 8 Procedure for calculating the next number in a sequence
Let $(d, d') \in P$ and a_i be a sorted number in a sequence. Then a_{i+1} is calculated as follows.

1. For all $d' \in \{1, ..., 9\}$: $D_{a_{i+1}}(d') = D_{a_i}(d)$.
2. For all $d' \in \{1, ..., 9\}$: $C_{a_{i+1}}(d') = C_{a_i}(d) + x(type(a_i), d')$.
3. Every stretch variable L_t should satisfy $\mathrm{LB}_{a_{i+1}}(L_t) + \mathrm{minC}_{a_{i+1}}(L_t) \geq 1$. Otherwise, the conditions are imposed on the lower bounds of the stretch variables in a_{i+1}, or the stretch variables in a_i are fixed to the smaller value and the calculation is restarted from a_i.

   Suppose $L_1, ..., L_u$ are such stretch variables. That is, $\mathrm{LB}_{a_i}(L_t) + \mathrm{minC}_{a_{i+1}}(L_t) < 1$ $(1 \leq t \leq u)$. Then the split subsets of possible assignments are:

   (1) in case every L_t comes to satisfy that $\mathrm{LB}_{a_{i+1}}(L_t) = 1 - \mathrm{minC}_{a_{i+1}}(L_t)$:
   a subset of possible assignments S_1 is formed.

   (2) in case one or more (at most u) L_ts in a_i are fixed and the lower bounds of the remaining variables (if any) are increased:
   For every $L_{t'}$ to be fixed, the possible values to be fixed range from $\mathrm{LB}_{a_i}(L_{t'})$ to $-\mathrm{minC}_{a_{i+1}}(L_{t'})$. For the remaining variable $L_{t''}$, $\mathrm{LB}_{a_{i+1}}(L_{t''}) = 1 - \mathrm{minC}_{a_{i+1}}(L_{t''})$. Consequently, one or more subsets of possible assignments $S_2, ..., S_{n_s}$ are formed. The calculation is restarted from a_i.

   The set of possible assignments S in a_i is partitioned into n_s subsets. That is, $\bigcup_{k=1}^{n_s} S_k = S$ (disjoint union). Note that some S_k may be $\emptyset$ because some L_t is actually a linear formula of stretch variables and is unable to have the required value. For each S_k, a_{i+1} that has the corresponding set of possible instances is formed.

◇

Remark 2 As shown in Definition 8, all the stretch variables only cycle in the stretch parts, i.e., $1 \to 2 \to 4 \to 8 \to 7 \to 5 \to 1$, $3 \to 6 \to 3$, or $9 \to 9$. There is no creation of a stretch variable in the consecutive calculation of the next numbers. Therefore, mapping of assignments to instances does not change.

3.2 Cyclic sequences and inductive sequences

We give the cyclic properties of a sequence of instances and a sequence of sets of instances.

Lemma 3 Let $\{a_n\}$ be a two-times sorting sequence, and $\{b_n\}$ be a sequence that consists of each instance b_n of a sorted number a_n under a certain assignment. Suppose that $b_i = b_{i+\ell}$ for some i and $\ell > 0$. Then, $b_i = b_{i+\ell} = b_{i+2\ell} = \cdots$. □

Lemma 4 Let $\{a_n\}$ be a two-times sorting sequence. Suppose that $a_i \equiv a_{i+\ell}$ and mapping of assignments to instances does not change from a_i to $a_{i+\ell}$ (i.e., there is no induction between a_i and $a_{i+\ell}$). Then, $a_i \equiv a_{i+\ell} \equiv a_{i+2\ell} \equiv \cdots$. □

There is the other relation than cyclic in the two-times sorting sequences.

Definition 9 A two-times sorting sequence $\{a_n\}$ is called *inductive* if there exist i and j with $i < j$ such that
(1) a_i and a_j are of the same type,
(2) there exist (at most 2 *) *decreasing digit(s)* d_1^{dec} (and d_2^{dec}) with the same non-zero stretch parts, i.e., $C_{a_i}(d_1^{\mathrm{dec}}) > C_{a_j}(d_1^{\mathrm{dec}})$ and $D_{a_i}(d_1^{\mathrm{dec}}) = D_{a_j}(d_1^{\mathrm{dec}}) =: L_{t_1^{\mathrm{dec}}} > 0$,
(3) there exist (at most 2 *) *increasing digit(s)* d_1^{inc} (and d_2^{inc}) with the same stretch parts, i.e.,

$C_{a_i}(d_1^{\mathrm{inc}}) < C_{a_j}(d_1^{\mathrm{inc}})$ and $D_{a_i}(d_1^{\mathrm{inc}}) = D_{a_j}(d_1^{\mathrm{inc}})$,
(4) the other digits are the same, i.e., for all $d \in \{1,...,9\}\backslash\{d_1^{\mathrm{dec}}, d_2^{\mathrm{dec}}, d_1^{\mathrm{inc}}, d_2^{\mathrm{inc}}\}$: $C_{a_i}(d) = C_{a_j}(d)$ and $D_{a_i}(d) = D_{a_j}(d)$,
(5) The sequence passes the induction check (See below).
The inductive relation is denoted by $a_i \approx a_j$. An *inductive length* ℓ^{ind} is defined as $j - i$. The numbers $\Delta(d_1^{\mathrm{dec}}) := C_{a_j}(d_1^{\mathrm{dec}}) - C_{a_i}(d_1^{\mathrm{dec}})$ and $\Delta(d_1^{\mathrm{inc}}) := C_{a_j}(d_1^{\mathrm{inc}}) - C_{a_i}(d_1^{\mathrm{inc}})$ are defined as a *decreasing step* and an *increasing step*, respectively. ◇

The induction check examines if the found relation is generally inductive or not. As for the detailed algorithm, refer to the full paper. If the check succeeds, a_i and a_j are really inductive. Since the calculation of the next number depends only on the type of a number (See Lemma 2 and Remark 1), the similar sequence from a_{i+1} to a_j is repeated as long as the types of numbers do not change. Then the inductions of decreasing digits and increasing digits hold. The numbers examined by the induction check are so general that they include all the numbers after a_j during the induction. The induction cycle is repeated until one of decreasing digits becomes one of the smallest values. No splits of the sequence occur except for the first cycle of the induction. If the check fails, and a_i and a_j are not indeed inductive.

We call the number after the induction as the *final induction number* or *fin*, which is calculated by the procedure of built-in induction. As for the detailed algorithm, refer to the full paper. Actually, the calculation should be restarted from the number one cycle before a final induction number in order to check a cycle related to the numbers in the last induction cycle. We call it the *pre-final induction number* or *pre-fin*. Note that the final induction number and the pre-final induction number respectively correspond to a_i and a_j when the count of induction cycles is 1.

Example 4 Let a_0 be 6...6 with $\ell_{a_0}(6) = L_1$ ($L_1 \geq 5$). Then,

$$
\begin{array}{lcl}
a_4 & = & 46.....689 = (\ 0\ \ 0\ \ 0\ \ \ \ 1\ \ 0\ \ L_1{-}2\ \ 0\ \ 1\ \ 1\ ), \\
a_5 & = & 3.....3789 = (\ 0\ \ 0\ \ L_1{-}2\ \ 0\ \ 0\ \ 0\ \ \ \ 1\ \ 1\ \ 1\ ), \\
a_6 & = & 56....6778 = (\ 0\ \ 0\ \ 0\ \ \ \ 0\ \ 1\ \ L_1{-}3\ \ 2\ \ 1\ \ 0\ ), \\
a_7 & = & 113....3556 = (\ 2\ \ 0\ \ L_1{-}3\ \ 0\ \ 2\ \ 1\ \ \ \ 0\ \ 0\ \ 0\ ), \\
a_8 & = & 112226...67 = (\ 2\ \ 3\ \ 0\ \ \ \ 0\ \ 0\ \ L_1{-}4\ \ 1\ \ 0\ \ 0\ ), \\
a_9 & = & 223...34445 = (\ 0\ \ 2\ \ L_1{-}4\ \ 3\ \ 1\ \ 0\ \ \ \ 0\ \ 0\ \ 0\ ), \\
a_{10} & = & 446...6889 = (\ 0\ \ 0\ \ 0\ \ \ \ 2\ \ 0\ \ L_1{-}4\ \ 0\ \ 2\ \ 1\ ).
\end{array}
$$

Here we see a_4 and a_{10} have the same type 4689. Comparing a_{10} with a_4, the digit 6 is decreasing with $D_{a_4}(6) = D_{a_{10}}(6) = L_1$, $\Delta(6) = -2$. $\mathrm{minC}_{a_4}^{a_{10}}(L_1) = C_{a_{10}}(6) = -4$, $\mathrm{minC}_{a_4}^{a_{10}}(L_1) + \mathrm{LB}_{a_{10}}(L_1) = -4 + 5 = 1 \geq 1$. The digits 4 and 8 are increasing with $\Delta(4) = 1$, $\Delta(8) = 1$. For all other digits, $\ell_{a_4}(d) = \ell_{a_{10}}(d)$. Therefore, the above sequence satisfies the conditions from (1) to (4) in Definition 9. Then the induction check is performed with using from a_4 to a_{10}. The induction check starts from a number $b_0 = 4...46...68...89 = (\ 0\ \ 0\ \ 0\ \ L_2\ \ 0\ \ L_1\ \ 0\ \ L_2\ \ 1\ )$. After 6 steps in a sequence, b_6 should be $(\ 0\ \ 0\ \ 0\ \ L_2+1\ \ 0\ \ L_1-2\ \ 0\ \ L_2+1\ \ 1\ )$. From b_0 to b_6, no splits should occur under the condition obtained from a_{10}, that is, $L_1 \geq 3$† and $L_2 \geq 2$.

The check succeeds. Then the sequence is inductive with $a_4 \approx a_{10}$. The inductive length is 6. That means that for every $i = 6k + 4$ $\ell_{a_i}(6) = L_1 - 2k - 2$ and $\ell_{a_i}(4) = \ell_{a_i}(8) = k + 1$ ($0 \leq k \leq \lfloor (L_1 - 3)/2 \rfloor$). The induction count is represented as a formula $y_1 = (L_1 - 3)/2$ if L_1 is odd, or $y_2 = (L_1 - 4)/2$ if L_1 is even. The minimum induction count is $\min(y_1) = \min(y_2) = 1$. Therefore, the final number on this induction would be either

$a_1^{\mathrm{fin}} = 4...468...89 = (\ 0\ \ 0\ \ 0\ \ L_1'\ \ 0\ \ 1\ \ 0\ \ L_1'\ \ 1\ )$ $L_1' = (L_1 - 1)/2 \geq 2$ if L_1 is odd, or
$a_2^{\mathrm{fin}} = 4...4668...89 = (\ 0\ \ 0\ \ 0\ \ L_1'\ \ 0\ \ 2\ \ 0\ \ L_1'\ \ 1\ )$ $L_1' = (L_1 - 2)/2 \geq 2$ if L_1 is even.

In the enumeration described below, the pre-final induction numbers are used instead of the final induction numbers. These are

$a_1^{\text{pre-fin}} = (\ 0\ \ 0\ \ 0\ \ L_1'\ \ 0\ \ 3\ \ 0\ \ L_1'\ \ 1\ )$ $L_1' = (L_1 - 3)/2 \geq 1$ if L_1 is odd, or
$a_2^{\text{pre-fin}} = (\ 0\ \ 0\ \ 0\ \ L_1'\ \ 0\ \ 4\ \ 0\ \ L_1'\ \ 1\ )$ $L_1' = (L_1 - 4)/2 \geq 1$ if L_1 is even. ◁

*These facts were confirmed by our early program.
†Note that $-\Delta(6) = 2$ is added to $\mathrm{LB}_{b_0}(L_1)$ in order to check the next induction cycle after a_{10}.

3.3 Procedure for the enumeration

In order to deal with the inductive relation, we have made a routine for the built-in induction. The routine first checks if the found induction is general. If the sequence is not indeed inductive, the routine is stopped and the enumeration is continued. Otherwise, the routine calculates one or more pre-final numbers, which are considered as the numbers after the induction. The enumeration is restarted from these numbers.

The enumeration proceeds in a depth-first manner. The split occurs when there is more than one possibility (1) whether some stretch variable is greater than a certain value or is fixed to the smaller value, and (2) when the built-in induction gives more than one number. Thus, the enumeration has a tree structure of sequences. With the split of a sequence, the conditions imposed on the sequence are accumulated as the conjunction. Generally, the conditions imposed later in a sequence are more strict or detailed than the earlier ones. Since our program does not check the consistency of the accumulated conditions, the enumeration tree may include the unsatisfiable subtrees in places.

The enumeration is terminated when a cyclic sequence is found. A cycle can be recognized as a terminal node in an enumeration tree. The program detects a cycle by checking the equivalence of two sorted numbers in a sequence with going backward up to the restarted number of the previous induction or the initial number (See Lemma 4).

In case of the split whether some stretch variable is greater than a certain value or is fixed to the smaller value, the condition is applied retroactively with going backward in the sequence, up to the restarted number of the previous induction or the initial number. The enumeration tree branches retroactively at the restarted number of the previous induction or the initial number.

On the other hand, in case of the split when the built-in induction gives more than one number, the enumeration tree branches just at the restarted number of the induction. This is reasonable because there exist no cycles between the numbers before an induction and the numbers after the induction (which was confirmed by the outputs of the program; See Section 4).

Figure 1 shows some typical types of the cycles. In the figure, the cyclic relation and the inductive relation are denoted by an arc with an arrow and by an arc with a star, respectively. The type A is the simplest case with no induction. The type B means that a simple cycle is found after one induction. The type C means that the found cycle starts from a number in the last induction cycle. Other types of cycles may be possible. The type D and E indicate that a cycle starts from a number in the middle of induction cycles and a cycle includes the induction cycles, respectively. Though our program can detect the cycles of type A, B and C, it cannot detect the cycles of type D and E. If such cycles exist in the enumeration trees, the program would terminate abnormally.

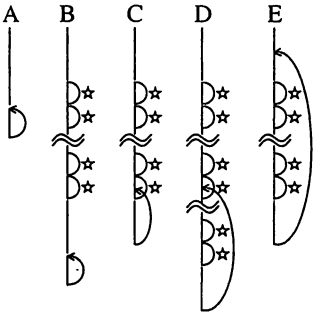

Figure 1: Illustration of the typical types of possible cycles.

3.4 An example of the enumeration trees

Example 5 Figure 2 shows the enumeration tree starting from $a_0 = 6...6$ with $\ell_{a_0}(6) = L_1$ ($L_1 \geq 1$), which is one of the 511 cases. The first split occurs at the first step. If $L_1 = 1$, a_1 is a particular number 12, which falls onto a cycle with the cyclic length 4 from a_4. Otherwise ($L_1 \geq 2$), a_1 is a general number $(\ 1\ \ 1\ \ L_1-1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ )$, which is followed by $a_2 = (\ 0\ \ 1\ \ 0\ \ 1\ \ 0\ \ L_1-1\ \ 0\ \ 0\ \ 0\ )$.

Thus, the enumeration proceeds with the branches at a_3, a_6, and a_8. Then, the inductive relation between a_4 and a_{10} is found. There are two possible pre-final induction numbers.

If L_1 is odd, the pre-fin is $(\ 0\ \ 0\ \ 0\ \ L_1'\ \ 0\ \ 3\ \ 0\ \ L_1'\ \ 1\ )$ $L_1' = (L_1 - 3)/2$, $L_1' \geq 1$. The enumeration is continued from this number as a_{11}. Here the difference between 11 and the actual number of steps considering the steps of induction is calculated as a formula and stored. Next,

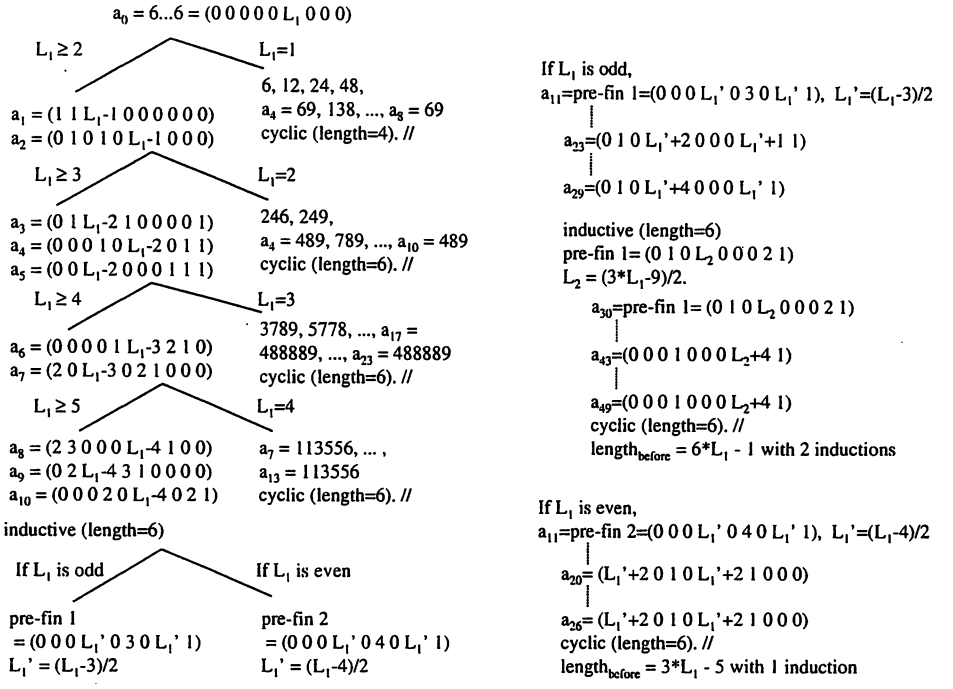

Figure 2: The enumeration tree starting from $a_0 = 6...6$.

another inductive relation between a_{23} and a_{29} is found. The only possible pre-fin of the induction is (0 1 0 L_2 0 0 0 2 1) $L_2 = (3L_1 - 9)/2$, $L_2 \geq 3$. The enumeration is continued from this number as a_{30}. Finally, the sequence falls onto a cycle where $a_{43} = a_{49}$ after two times of induction. The length before cyclic is represented as a linear formula $6L_1 - 1$.

On the other hand, if L_1 is even, the pre-fin is (0 0 0 L_1' 0 4 0 L_1' 1) $L_1' = (L_1 - 4)/2$, $L_1' \geq 1$. The enumeration is continued from this number and the cyclic relation $a_{20} = a_{26}$ is found. The length before cyclic is $3L_1 - 5$. ◁

3.5 Discussion on the degenerated cases and particular cases

We need to consider the cases in which some stretch variables satisfy a set of linear equations.

Definition 10 Let m_1 and m_2 be sorted numbers. Suppose that $C_{m_1}(d) = C_{m_2}(d)$ and $D_{m_1}(d) = D_{m_2}(d)$ for all $d \in \{1, ..., 9\}$, and all stretch variables in m_1 are independent and free except for the lower bounds. A sorted number m_2 is said to be *degenerated* if for some d_1, ..., d_k such that $D_{m_2}(d_i) = L_{t_i} > 0$, a set of linear equations $E = \{ e(L_{t_1}, ..., L_{t_k}) = 0 \}$ holds.
In other words, m_1 covers a degenerated m_2, or m_2 is a degenerated case of m_1.
Especially, if $E = \{L_{t_1} = c_{t_1}, ..., L_{t_k} = c_{t_k}\}$ holds, m_2 is called a *particular* case of m_1. ◇

Lemma 5 The degenerated or particular case of a sorted number m is less general than m. □

Our program only checks the simple cases such as $L_{t_1} = L_{t_2}$, etc. By checking this type of degeneration, the sequences of which cyclic length is 1 have been found. That is sufficient for our purpose.

4 Proof of Theorem 1

The outputs of our program have shown that all two-times sorting sequences result in the cycles of which lengths are 1, 2, 3, 4, 5, 6, or 12. That means there are no cycles of the type D or E in

Figure 1 in the enumeration. We can give an example sequence of which cyclic length ℓ is 1, 2, 3, 4, 5, 6, or 12, respectively.

$\ell = 1$: $(L_1 \quad L_1 \quad 0 \quad L_1 \quad L_1 \quad 0 \quad L_1 \quad L_1 \quad L_2)$ $L_1, L_2 \geq 1$,
$\ell = 2$: $(2 \quad 1 \quad 0 \quad 2 \quad 1 \quad 0 \quad 2 \quad 1 \quad L_1)$ $L_1 \geq 1$,
$\ell = 3$: $(1 \quad 2 \quad 0 \quad 1 \quad 1 \quad 0 \quad 2 \quad 1 \quad L_1)$ $L_1 \geq 1$,
$\ell = 4$: $(1 \quad 0 \quad L_1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0)$ $L_1 \geq 1$,
$\ell = 5$: $9 \mapsto 18 \mapsto 36 \mapsto 27 \mapsto 45 \mapsto 9$ (virtually only one cycle with no general ones),
$\ell = 6$: $(0 \quad 0 \quad 3 \quad 1 \quad 0 \quad 0 \quad 0 \quad L_1 \quad 1)$ $L_1 \geq 1$,
$\ell = 12$: $(1 \quad 0 \quad 0 \quad 0 \quad 0 \quad L_1 \quad 1 \quad 0 \quad 0)$ $L_1 \geq 1$.

Since our program deals with the general cases, many particular instances are included in such general numbers. The fact that all the general sorted numbers fall onto cycles shows that all the sorted numbers including all particular instances fall onto the same cycles. In addition, though the enumeration trees may have the non-existent subtrees, we have found at least one example of which cyclic length is 1, 2, 3, 4, 5, 6, or 12, respectively.

However, there is still the possibility of existence of the particular cycles other than we have found. Here, we discuss the possible cycles that the degenerated cases or the particular cases may fall onto. We see that the set of possible cycles of general cases is {1, 2, 3, 4, 5, 6, 12}. Since the outputs cover all general cases that include any degenerated or particular case, the particular shorter cycles have to be within the found cycles. Therefore, if there exist some particular cycles, the cyclic lengths of those cycles have to be the divisors of those of the general cycles. Since the set of divisors of each element in the set {1, 2, 3, 4, 5, 6, 12} is also {1, 2, 3, 4, 5, 6, 12}, there are no other cyclic lengths than that.

5 Conclusion

The program outputs the length before cyclic as a constant if no inductions occur before reaching a terminal, or as a formula of some stretch variables if one or more inductions occur. We see that all formulae are linear on one or more stretch variables. The maximum number of variables in a formula is 5. Therefore, the length before cyclic is unbounded.

Heretofore, we only consider the sorted numbers. We can easily prove that any instance of any sorted number in the enumeration trees can be the second number of a sequence with starting from a certain natural number. For instance, if we want to form any instance of any sorted number m as the second number in a sequence, we can choose a natural number $n = (m \times 10)/2$ as the first number. Moreover, if we choose a natural number $n = (m \times 100)/2$ as the first number, it is not in the enumeration trees because it contains one or two zeros. Therefore, the number of steps for any natural number is at most the value for the 511 root cases plus one.

Our approach on this study mainly uses the enumeration of sequences by the computer. This approach will possibly be applicable to other integer sequences. Our program and its outputs (huge!) are available on the Web `http://caster.cs.inf.shizuoka.ac.jp/~sakuta/TTSS/`.

References

[1] Makoto Araya. On the two-times sorting sequence. In *Proceedings of the 42nd Programming Symposium*, pages 45–48, January 2001. (in Japanese).

[2] Zahidul Haque, Makoto Araya, and Hiroyuki Iida. Two-times sorting. In Hiromasa Nakatani and Hiroyuki Iida, editors, *Proceedings of the Joint International Conference on Advanced Science and Technology (JICAST 2000)*, pages 133–136, Shizuoka University, 2000.

[3] Hiroyuki Iida. Reports on GPCC2000 challenges and results. In *Proceedings of the 42nd Programming Symposium*, pages 43–44, January 2001. (in Japanese).

[4] Neil J. A. Sloane. The On-Line Encyclopedia of Integer Sequences. published electronically at http://www.research.att.com/~njas/sequences/, 2001.