Efficient Algorithms for a Generalized Shuffling Problem

Daxin Zhu, Lei Wang, Jun Tian* and Xiaodong Wang*

Abstract—**In this paper we study a set of generalized shuffling problems. The goal of each problem is to make a smallest number of moves to reach the final state of the problem from its initial state. We are interested in algorithms which, given integers n, generate the corresponding move sequences to reach the final state of the game with smallest number of steps. In this paper we present the optimal algorithms to generate the optimal move sequences of the problems consisting of n black coins and n white coins, and finally, we present the explicit solutions for all of the generalized shuffling problems of size n.**

Index Terms—**Computer games, shuffling problems, optimal algorithms, explicit solutions.**

I. Introduction

The shuffling problem is actually a single player computer game similar to the moving coins puzzle [2], [3], [9], which is played by rearranging one configuration of unit disks in the plane into another configuration by a sequence of moves, each repositioning a coin in an empty cell that touches at least two other coins. In our shuffling problem, there are two types of objects, black coins b and white coins w. A sequence of $2n+2$ cells numbered $0,1,\cdots,2n-1$, and n black coins and n white coins are given. The $2n$ coins are put on the cells from left to right in a row. Initially, the n black coins are put on the cells $0,1,\cdots,n-1$, and the n white coins are put on the cells $n,n-1,\cdots,2n-1$. The rightmost two cells $2n$ and $2n+1$ are vacant, denoted as O. In the final state of the game, the cells of even number $2,4,\cdots,2n$ are occupied by white coins, and the cells of odd number $3,5,\cdots,2n+1$ are occupied by black coins, leaving the two cells 0 and 1 vacant.

Manuscript received August 18, 2014; revised October 10, 2014. This work was supported in part by the Natural Science Foundation of Fujian under Grant No.2013J01247, Fujian Provincial Key Laboratory of Data-Intensive Computing and Fujian University Laboratory of Intelligent Computing and Information Processing.

Daxin Zhu is with Quanzhou Normal University, Quanzhou, China.(email:dex@qztc.edu.cn)

Lei Wang is with Microsoft AdCenter, Bellevue, WA 98004, USA.(email:wanglei07@gmail.com)

Jun Tian is with Fujian Medical University, Fuzhou, China.(email:tianjunfjmu@126.com)

Xiaodong Wang is with Fujian University of Technology, Fuzhou, China.(email:wangxd135@139.com)

*Corresponding author.

A move of the game consists of shifting two adjacent coins, keeping their order, into the current two vacant cells. The shuffling problem is to make a smallest number of moves to rearrange the initial state $\overbrace{b\cdots b}^{n}\overbrace{w\cdots w}^{n}OO$ to the final state $OO\overbrace{wb\cdots wb}^{2n}$. This original shuffling problem can be denoted as $S(n,n,r)$, because the vacant cells are located rightmost in its initial state. The problem $S(n,n,l)$ is a variant of $S(n,n,r)$. The problem $S(n,n,l)$ is to rearrange its initial state $OO\overbrace{b\cdots b}^{n}\overbrace{w\cdots w}^{n}$ to its final state $\overbrace{wb\cdots wb}^{2n}OO$. Another variant of $S(n,n,r)$ is its inverse problem $S^{-1}(n,n,r)$. The inverse problem is to rearrange inversely from $OO\overbrace{wb\cdots wb}^{2n}$ to $\overbrace{b\cdots b}^{n}\overbrace{w\cdots w}^{n}OO$. Similarly, the inverse problem $S^{-1}(n,n,l)$ is to rearrange inversely from $\overbrace{wb\cdots wb}^{2n}OO$ to $OO\overbrace{b\cdots b}^{n}\overbrace{w\cdots w}^{n}$.

In this paper, we will discuss more new variants of the problem of $S(n,n,r)$ and their inverse problems. These new variants of $S(n,n,r)$ are $S(n,n-1,r)$, $S(n-1,n,r)$, $S(n,n-1,l)$ and $S(n-1,n,l)$. The definitions of these new variants of $S(n,n,r)$ are similar to the original shuffling problem. For example, the problem $S(n,n-1,r)$ is to rearrange its initial state $\overbrace{b\cdots b}^{n}\overbrace{w\cdots w}^{n-1}OO$ to its final state $OO\overbrace{bwb\cdots bwb}^{2n-1}$. The others are defined similarly, as shown in Table 1.

In a recent paper [9], an optimal algorithm to generate an optimal move sequence of the problem $S(n,n,r)$ was presented. Based on the optimal algorithm of the problem $S(n,n,r)$, we can build new optimal algorithms for all the new variants of the shuffling problem listed above in this paper.

This paper is structured as follows.

In the following 4 sections we describe our new algorithms for the generalized shuffling problems. In section 2 we prove the lower bounds of the generalized shuffling problems. The optimal recursive construction algorithms are proposed in section 3. Based on the recursive algorithm proposed in section 3, the explicit solutions for the optimal move sequence of the general problems are presented in section

Table I
THE GENERALIZED SHUFFLING PROBLEMS

Problem	Initial State	Final State
$S(n,n,r)$	$\overbrace{b\cdots b}^{n}\overbrace{w\cdots w}^{n}OO$	$OO\overbrace{wb\cdots wb}^{2n}$
$S(n,n,l)$	$OO\overbrace{b\cdots b}^{n}\overbrace{w\cdots w}^{n}$	$\overbrace{wb\cdots wb}^{2n}OO$
$S^{-1}(n,n,r)$	$OO\overbrace{wb\cdots wb}^{2n}$	$\overbrace{b\cdots b}^{n}\overbrace{w\cdots w}^{n}OO$
$S^{-1}(n,n,l)$	$\overbrace{wb\cdots wb}^{2n}OO$	$OO\overbrace{b\cdots b}^{n}\overbrace{w\cdots w}^{n}$
$S(n,n-1,r)$	$\overbrace{b\cdots b}^{n}\overbrace{w\cdots w}^{n-1}OO$	$OO\overbrace{bwb\cdots bwb}^{2n-1}$
$S(n-1,n,r)$	$\overbrace{w\cdots w}^{n-1}\overbrace{b\cdots b}^{n}OO$	$OO\overbrace{bwb\cdots bwb}^{2n-1}$
$S(n,n-1,l)$	$OO\overbrace{b\cdots b}^{n}\overbrace{w\cdots w}^{n-1}$	$\overbrace{bwb\cdots bwb}^{2n-1}OO$
$S(n-1,n,l)$	$OO\overbrace{w\cdots w}^{n-1}\overbrace{b\cdots b}^{n}$	$\overbrace{bwb\cdots bwb}^{2n-1}OO$
$S^{-1}(n,n-1,r)$	$OO\overbrace{bwb\cdots bwb}^{2n-1}$	$\overbrace{b\cdots b}^{n}\overbrace{w\cdots w}^{n-1}OO$
$S^{-1}(n-1,n,r)$	$OO\overbrace{bwb\cdots bwb}^{2n-1}$	$\overbrace{w\cdots w}^{n-1}\overbrace{b\cdots b}^{n}OO$
$S^{-1}(n,n-1,l)$	$\overbrace{bwb\cdots bwb}^{2n-1}OO$	$OO\overbrace{b\cdots b}^{n}\overbrace{w\cdots w}^{n-1}$
$S^{-1}(n-1,n,l)$	$\overbrace{bwb\cdots bwb}^{2n-1}OO$	$OO\overbrace{w\cdots w}^{n-1}\overbrace{b\cdots b}^{n}$

4. Some concluding remarks are in section 5.

II. The Lower Bounds of the Problems

It has been proved that the problems $S(n,n,r)$, $S(n,n,l)$, $S^{-1}(n,n,r)$, and $S^{-1}(n,n,l)$ cannot be solved in fewer than n moves [1], [6], [7]. Similar results for the problems $S(n,n-1,r)$, $S(n,n-1,l)$, $S(n-1,n,r)$, $S(n-1,n,l)$, $S^{-1}(n,n-1,r)$, $S^{-1}(n,n-1,l)$,$S^{-1}(n-1,n,r)$,and $S^{-1}(n-1,n,l)$ are also true.

Theorem 1: The problems $S(n,n-1,r)$, $S(n,n-1,l)$, $S(n-1,n,r)$, $S(n-1,n,l)$, $S^{-1}(n,n-1,r)$, $S^{-1}(n,n-1,l)$,$S^{-1}(n-1,n,r)$,and $S^{-1}(n-1,n,l)$ cannot be solved in fewer than n moves.

Proof. We consider the problem $S(n,n-1,r)$ first.

we count the number of variations, bw and wb, similarly to the counting of changes of sign in Descarte's rule of signs. In the initial state of the problem $S(n,n-1,r)$, there is only one variation. But, In the final state of the problem $S(n,n-1,r)$, there are total $2n-2$ variations. It is readily to see that the variations are increased by $2n-3$ from initial state to the final state of the problem $S(n,n-1,r)$. In the first step of move, at most one variation can be added and in the subsequent moves at most two variations can be added in each step except the last step. We have noticed that if the problem is solved, the last step of move must be removing the two adjacent coins xy located at cells 0 and 1 to the current vacant cells located at cells i and $i+1$, $2 \le i < 2n$. There are two cases of the two adjacent coins xy to be distinguished. It is readily seen that if $xy = bw$ then the last move produces no increment in variations. In the case of $xy = wb$, exactly one variation will be added by the last step of move. Therefore, at most one variation can be added in the last step.

If the final state is reached after m steps, then at most $2(m-2)+2 = 2m-2$ variations are added. Therefore, we have, $2m-2 \ge 2n-3$ and so $m \ge n-1/2$. Since m is an integer, we have $m \ge n$.

In other words, it needs at least n steps to reach the final state of $S(n,n-1,r)$ from its initial state.

The proofs for the other 7 problems are similar. □

III. The Optimal Recursive Algorithms

In the following discussion, we will use a vector $x = (x_1, x_2, \cdots, x_n)$ to denote the solution of the problem. For $i = 1, 2, \cdots, n$, the move of step i is x_i. This means that in the move of step i, we move the adjacent pair of coins located at cells x_i and x_i+1 to the current vacant cells and leaving the cells x_i and x_i+1 the new vacant cells.

A. The Solutions for the Problems of Type (n,n)

1) The Solution for the Problem $S(n,n,r)$: We discuss the problem $S(n,n,r)$ first. It is not difficult to generate all optimal solutions of the problem with small size by a backtracking algorithm.

For example, in the cases of $n = 4,5,6,7$, the corresponding optimal solutions are $x = (1,4,7,0)$, $(1,7,4,9,0)$, $(1,7,3,8,11,0)$ and $(1,10,4,9,6,13,0)$ respectively.

The Decrease-and-Conquer strategy [4], [5] for algorithm design can be exploited to design the optimal recursive algorithms for our purpose. For the cases of $n \ge 8$, we can find one optimal solution for the game recursively as follows.

We first make 2 moves $x_1 = 1$ and $x_2 = 2n-4$. The status is changed to

$$bwwb\overbrace{b\cdots b}^{n-4}\overbrace{w\cdots w}^{n-4}OOwwbb$$

Now, the coins located at the cells $4, 5, \cdots, 2n-3$ are exactly the same as the initial state of the problem $S(n-4, n-4, r)$. We can use our recursive algorithm to this subproblem to get

$$bwwbOO\overbrace{wb\cdots wb}^{2n-8}wwbb$$

We finally make 2 moves $x_{n-1} = 2n-1$ and $x_n = 0$ to get the final status

$$OO\overbrace{wb\cdots wb}^{2n}$$

The recursive algorithm can be described as follows.

Algorithm III.1: SNNR($first, k$)

```
if $k < 8$
  then construct the solution directly
  else
    comment: first 2 moves
    $x_i \leftarrow first + 1$
    $x_{i+1} \leftarrow first + 2 * k - 4$
    $i \leftarrow i + 2$
    comment: recursive moves
    SNNR($first + 4, k - 4$)
    comment: last 2 moves
    $x_i \leftarrow first + 2 * k - 1$
    $x_{i+1} \leftarrow first$
    $i \leftarrow i + 2$
```

In the algorithm described above, the parameters $first$ and k describe the initial state of the subproblem of size k starting at cell $first$. The current step of move is stored in a global variable i which is initialized with 1.

It is obviously that the above algorithm uses n moves to get the the final status. The algorithm is an optimal algorithm, since n is a lower bound of the problem $S(n, n, r)$.

2) The Solution for the Problem $S(n, n, l)$: The problem $S(n, n, l)$ is symmetric to the problem $S(n, n, r)$. The solutions for the problem in the cases of $n = 4, 5, 6, 7$, are $x = (7, 4, 1, 8)$, $(9, 3, 6, 1, 10)$, $(11, 5, 9, 4, 1, 12)$ and $(13, 4, 10, 5, 8, 1, 14)$ respectively. It can be verified that if $y = (y_1, \cdots, y_n)$ is an optimal solution for the problem $S(n, n, r)$, then $x = (2n-y_1, \cdots, 2n-y_n)$ is also an optimal solution for the problem $S(n, n, l)$.

For the cases of $n \geq 8$, the optimal solution for the problem can also be found recursively.

We first make 2 moves $x_1 = 2n - 1$ and $x_2 = 4$. The status is changed to

$$wwbbOO\overbrace{b\cdots b}^{n-4}\overbrace{w\cdots w}^{n-4}wbbw$$

Then the subproblem $S(n-4, n-4, r)$ is solved recursively to get

$$wwbb\overbrace{wb\cdots wb}^{2n-8}OOwbbw$$

We finally make 2 moves $x_{n-1} = 1$ and $x_n = 2n$ to get the final status

$$\overbrace{wb\cdots wb}^{2n}OO$$

The recursive algorithm can be described as follows.

Algorithm III.2: SNNL($first, k$)

```
if $k < 8$
  then construct the solution directly
  else
    comment: first 2 moves
    $x_i \leftarrow first + 2 * k - 1$
    $x_{i+1} \leftarrow first + 4$
    $i \leftarrow i + 2$
    comment: recursive moves
    SNNL($first + 4, k - 4$)
    comment: last 2 moves
    $x_i \leftarrow first + 1$
    $x_{i+1} \leftarrow first + 2 * k$
    $i \leftarrow i + 2$
```

It is obviously that the above algorithm uses n moves to get the the final status. The algorithm is an optimal algorithm, since n is a lower bound of the problem $S(n, n, l)$.

3) The Solutions for the Problems $S^{-1}(n, n, r)$ and $S^{-1}(n, n, l)$: The problem $S^{-1}(n, n, l)$ is an inverse problem of $S(n, n, l)$. For the cases of $n \geq 8$, the optimal solution for the problem can also be found recursively.

We first make 2 moves $x_1 = 1$ and $x_2 = 2n - 4$. The status is changed to

$$wwbb\overbrace{wb\cdots wb}^{2n-8}OOwbbw$$

Then the subproblem $S^{-1}(n-4, n-4, l)$ is solved recursively to get

$$wwbbOO\overbrace{b\cdots b}^{n-4}\overbrace{w\cdots w}^{n-4}wbbw$$

We finally make 2 moves $x_{n-1} = 2n - 1$ and $x_n = 0$ to get the final status

$$OO\overbrace{b\cdots b}^{n}\overbrace{w\cdots w}^{n}$$

Compared to the solution for the problem $S(n, n, r)$, we have found that the solution for the problem $S^{-1}(n, n, l)$ is exactly the same as the solution for the problem $S(n, n, r)$. Similarly, we can conclude that the solution for the problem $S^{-1}(n, n, r)$ is exactly the same as the solution for the problem $S(n, n, l)$.

B. The Solutions for the Problems of Type $(n, n-1)$

1) The Solution for the Problem $S(n, n-1, r)$: The problem $S(n, n-1, r)$ can be reduced to the problem $S(n, n, l)$ as follows.

The solutions for the problem in the cases of $n = 3, 4, 5, 6$, are $x = (1, 4, 0)$, $(0, 3, 6, 0)$, $(2, 6, 1, 8, 0)$, and $(1, 7, 2, 5, 10, 0)$ respectively.

For the cases of $n \geq 7$, we first make 1 move $x_1 = 1$. The initial status $\overbrace{b \cdots b}^{n}\overbrace{w \cdots w}^{n-1}OO$ is changed to

$$bOO\overbrace{b \cdots b}^{n-3}\overbrace{w \cdots w}^{n-3}wwbb$$

Then the subproblem $S(n-3, n-3, r)$ is solved to get

$$b\overbrace{wb \cdots wb}^{2n-6}OOwwbb$$

We finally make 2 moves $x_{n-1} = 2n-2$ and $x_n = 0$ to get the final status $OO\overbrace{bwb \cdots bwb}^{2n-1}$.

The algorithm can be described as follows.

```
Algorithm III.3: SNN1R(n)

if $n < 8$
  then construct the solution directly
  else {
    comment: first move
    $x_1 \leftarrow 1$
    comment: reduced to $S(n, n, l)$
    SNNL$(1, n-3)$
    comment: last 2 moves
    $x_{n-1} \leftarrow 2n-2$
    $x_n \leftarrow 0$
  }
```

Since the algorithm SNNL$(1, n-3)$ needs $n-3$ moves, the algorithm SNN1R(n) requires n moves.

2) The Solution for the Problem $S(n-1, n, r)$: The problem $S(n-1, n, r)$ can be reduced to the problem $S(n, n, r)$ as follows.

The solutions for the problem in the cases of $n = 3, 5, 6$, are $x = (1, 3, 0)$, $(6, 1, 8, 5, 0)$, and $(1, 8, 1, 4, 7, 0)$ respectively.

For the cases of $n \geq 7$, we first make 2 moves $x_1 = n-2$ and $x_2 = 2n-4$. The initial status $\overbrace{w \cdots w}^{n-1}\overbrace{b \cdots b}^{n}OO$ is changed to

$$\overbrace{w \cdots w}^{n-2}\overbrace{b \cdots b}^{n-2}OObwb$$

Then the subproblem $S(n-2, n-2, r)$ is solved to get

$$OOb\overbrace{bw \cdots bw}^{2n-4}bwb$$

This is the final status $OO\overbrace{bwb \cdots bwb}^{2n-1}$.

The algorithm can be described as follows.

```
Algorithm III.4: SN1NR(n)

if $n < 6$
  then construct the solution directly
  else {
    comment: first 2 moves
    $x_1 \leftarrow n-2$
    $x_2 \leftarrow 2n-4$
    comment: reduced to $S(n, n, r)$
    SNNR$(0, n-2)$
  }
```

Since the algorithm SNNR$(0, n-2)$ needs $n-2$ moves, the algorithm SN1NR(n) requires n moves.

3) The Solution for the Problem $S(n-1, n, l)$: The problem $S(n-1, n, l)$ can be reduced to the problem $S(n, n, r)$ as follows.

The solutions for the problem in the cases of $n = 3, 4, 5, 6$, are $x = (4, 1, 5)$, $(7, 4, 1, 7)$, $(7, 3, 8, 1, 9)$ and $(10, 4, 9, 6, 1, 11)$ respectively.

For the cases of $n \geq 7$, we first make 1 move $x_1 = 2n-2$. The initial status $OO\overbrace{w \cdots w}^{n-1}\overbrace{b \cdots b}^{n}$ is changed to

$$bbww\overbrace{w \cdots w}^{n-3}\overbrace{b \cdots b}^{n-3}OOb$$

Then the subproblem $S(n-3, n-3, r)$ is solved to get

$$bbwwOO\overbrace{bw \cdots bw}^{2n-6}b$$

We finally make 2 moves $x_{n-1} = 1$ and $x_n = 2n-1$ to get the final status $\overbrace{bwb \cdots bwb}^{2n-1}OO$.

The algorithm can be described as follows.

```
Algorithm III.5: SN1NR(n)

if $n < 7$
  then construct the solution directly
  else {
    comment: first move
    $x_1 \leftarrow 2n-2$
    comment: reduced to $S(n, n, r)$
    SNNR$(4, n-3)$
    comment: last 2 moves
    $x_{n-1} \leftarrow 1$
    $x_n \leftarrow 2n-1$
  }
```

Since the algorithm SNNR$(4, n-3)$ needs $n-3$ moves, the algorithm SN1NR(n) requires n moves.

4) The Solution for the Problem $S(n, n-1, l)$: The problem $S(n, n-1, l)$ can be reduced to the problem $S(n, n, l)$ as follows.

The solutions for the problem in the cases of $n = 3, 5, 6$, are $x = (4, 2, 5)$, $(3, 8, 1, 4, 9)$ and $(10, 3, 0, 7, 4, 11)$ respectively.

For the cases of $n \geq 7$, we first make 2 moves $x_1 = n+1$ and $x_2 = 3$. The initial status $OO\overbrace{b\cdots b}^{n}\overbrace{w\cdots w}^{n-1}$ is changed to

$$bwbOO\overbrace{b\cdots b}^{n-2}\overbrace{w\cdots w}^{n-2}$$

Then the subproblem $S(n-3, n-3, l)$ is solved to get

$$bwb\overbrace{wb\cdots wb}^{2n-6}OO$$

This is the final status $\overbrace{bwb\cdots bwb}^{2n-1}OO$.

The algorithm can be described as follows.

```
Algorithm III.6: SN1NL($n$)
  if $n < 7$
    then construct the solution directly
    else {
      comment: first 2 moves
      $x_1 \leftarrow n+1$
      $x_2 \leftarrow 3$
      comment: reduced to $S(n, n, l)$
      SNNL($3, n-2$)
    }
```

Since the algorithm SNNL$(3, n-2)$ needs $n-2$ moves, the algorithm SN1NL(n) requires n moves.

5) The Solution for the Problem $S^{-1}(n, n-1, r)$: The problem $S^{-1}(n, n-1, r)$ can be reduced to the problem $S^{-1}(n, n, l)$ as follows.

The solutions for the problem in the cases of $n = 3, 4, 5, 6$, are $x = (4, 1, 5)$, $(3, 6, 0, 7)$, $(8, 1, 6, 2, 9)$ and $(2, 5, 10, 7, 1, 11)$ respectively.

For the cases of $n \geq 7$, we first make 2 moves $x_1 = 2n-2$ and $x_2 = 2n-5$. The initial status $OO\overbrace{bwb\cdots bwb}^{2n-1}$ is changed to

$$b\overbrace{wb\cdots wb}^{2n-6}OOwwbb$$

Then the subproblem $S^{-1}(n-3, n-3, l)$, which is equivalent to $S(n-3, n-3, r)$, is solved to get

$$bOO\overbrace{b\cdots b}^{n-3}\overbrace{w\cdots w}^{n-3}wwbb$$

We finally make 1 move $x_n = 2n-1$ to get the final status $\overbrace{b\cdots b}^{n}\overbrace{w\cdots w}^{n-1}OO$.

The algorithm can be described as follows.

```
Algorithm III.7: S$^{-1}$NN1R($n$)
  if $n < 7$
    then construct the solution directly
    else {
      comment: first 2 moves
      $x_1 \leftarrow 2n-2$
      $x_2 \leftarrow 2n-5$
      comment: reduced to $S^{-1}(n, n, l)$
      SNNR($1, n-3$)
      comment: last move
      $x_n \leftarrow 2n-1$
    }
```

Since the algorithm SNNR$(1, n-3)$ needs $n-3$ moves, the algorithm S^{-1}NN1R(n) requires n moves.

6) The Solution for the Problem $S^{-1}(n-1, n, r)$: The problem $S^{-1}(n-1, n, r)$ can be reduced to the problem $S^{-1}(n, n, r)$ as follows.

The solutions for the problem in the cases of $n = 3, 5, 6$, are $x = (3, 1, 5)$, $(5, 8, 1, 6, 9)$ and $(5, 8, 1, 6, 4, 11)$ respectively.

For the cases of $n \geq 7$, we first solve the subproblem $S^{-1}(n-2, n-2, r)$, which is equivalent to $S(n-2, n-2, l)$, for the first $2n-2$ cells to get

$$\overbrace{w\cdots w}^{n-2}\overbrace{b\cdots b}^{n-2}OObwb$$

We finally make 2 moves $x_{n-1} = n-2$ and $x_n = 2n-1$ to get the final status $\overbrace{w\cdots w}^{n-1}\overbrace{b\cdots b}^{n}OO$.

The algorithm can be described as follows.

```
Algorithm III.8: S$^{-1}$N1NR($n$)
  if $n < 7$
    then construct the solution directly
    else {
      comment: reduced to $S^{-1}(n, n, r)$
      SNNL($0, n-2$)
      comment: last 2 moves
      $x_{n-1} \leftarrow n-2$
      $x_n \leftarrow 2n-1$
    }
```

Since the algorithm SNNL$(0, n-2)$ needs $n-2$ moves, the algorithm S^{-1}N1NR(n) requires n moves.

7) The Solution for the Problem $S^{-1}(n-1,n,l)$: The problem $S^{-1}(n-1,n,l)$ can be reduced to the problem $S^{-1}(n,n,r)$ as follows.

The solutions for the problem in the cases of $n = 3,4,5,6$, are $x = (1,4,0)$, $(1,4,7,0)$, $(1,8,3,7,0)$ and $(1,6,9,4,10,0)$ respectively.

For the cases of $n \geq 7$, we first make 2 moves $x_1 = 1$ and $x_2 = 4$. The initial status $\overbrace{bwb\cdots bwb}^{2n-1}OO$ is changed to

$$bbwwOO\overbrace{bw\cdots bw}^{2n-6}b$$

Then the subproblem $S^{-1}(n-3,n-3,r)$, which is equivalent to $S(n-3,n-3,l)$, is solved to get

$$bbww\overbrace{w\cdots w}^{n-3}\overbrace{b\cdots b}^{n-3}OOb$$

We finally make 1 move $x_n = 0$ to get the final status $OO\overbrace{w\cdots w}^{n-1}\overbrace{b\cdots b}^{n}$.

The algorithm can be described as follows.

```
Algorithm III.9: S^{-1}N1NL(n)

if $n < 7$
  then construct the solution directly
  else { comment: first 2 moves
         $x_1 \leftarrow 1$
         $x_2 \leftarrow 4$
         comment: reduced to $S^{-1}(n,n,r)$
         SNNL$(4, n-3)$
         comment: last move
         $x_n \leftarrow 0$
```

Since the algorithm SNNL$(4,n-3)$ needs $n-3$ moves, the algorithm S^{-1}N1NL(n) requires n moves.

8) The Solution for the Problem $S^{-1}(n,n-1,l)$: The problem $S^{-1}(n,n-1,l)$ can be reduced to the problem $S^{-1}(n,n,l)$ as follows.

The solutions for the problem in the cases of $n = 3,5,6$, are $x = (2,4,0)$, $(4,1,8,3,0)$ and $(2,5,10,1,7,0)$ respectively.

For the cases of $n \geq 7$, we first solve the subproblem $S^{-1}(n-2,n-2,l)$, which is equivalent to $S(n-2,n-2,r)$, for the last $2n-2$ cells to get

$$bwb\overbrace{wb\cdots wb}^{2n-4}OO$$

We finally make 2 moves $x_{n-1} = n+1$ and $x_n = 0$ to get the final status $OO\overbrace{b\cdots b}^{n}\overbrace{w\cdots w}^{n-1}$.

Table II
SMALL TWO DIMENSIONAL ARRAY d

n \ i	1	2	3	4	5	6	7
4	1	4	7	0	0	0	0
5	1	7	4	9	0	0	0
6	1	7	3	8	11	0	0
7	1	10	4	9	6	13	0

The algorithm can be described as follows.

```
Algorithm III.10: S^{-1}NN1L(n)

if $n < 7$
  then construct the solution directly
  else { comment: reduced to $S^{-1}(n,n,l)$
         SNNR$(3, n-2)$
         comment: last 2 moves
         $x_{n-1} \leftarrow n+1$
         $x_n \leftarrow 0$
```

Since the algorithm SNNR$(3,n-2)$ needs $n-2$ moves, the algorithm S^{-1}NN1L(n) requires n moves.

IV. The Explicit Solutions of the Problems

A. The Explicit Solutions of the Problems of Type (n,n)

1) The Explicit Solutions of the Problem $S(n,n,r)$: In this section we will discuss the explicit expression of function x. We will discuss the solution for the problem $S(n,n,r)$ first.

For the small size cases of $n = 4,5,6,7$, the optimal solutions of the problem can be listed as a small two dimensional array d as follows.

For the problems of size $n = 4,5,6,7$, the step $i, 1 \leq i \leq n$ of the optimal solution can be expressed as $d(n,i)$. For the cases of the subproblems of size $n-t = 4,5,6,7$, starting at cell t, the corresponding step $j, 1 \leq j \leq n-t$ of the optimal solution can be expressed as $t + d(n-t,j)$. For the general cases of the subproblems of size n, starting at cells t, if the corresponding step $j, 1 \leq j \leq n$ of the optimal solution is denoted as $\alpha(n,t,j)$, then for the problem of size n, the optimal move $x_i, 1 \leq i \leq n$ must be $\alpha(n,0,i), 1 \leq i \leq n$.

According to the recursive algorithm presented in previous section, the function $\alpha(n,t,j)$ can be computed as follows.

$$\alpha(n,t,j)=\begin{cases} t+d(n-t,j) & 3<n-t \text{ and } n-t<8\\ t+1 & j=1\\ 2n-t-4 & j=2\\ 2n-t-1 & j=n-t-1\\ t & j=n-t\\ \alpha(n,t+4,j-2) & \text{otherwise}\end{cases} \tag{1}$$

If i or $n-i$ is a constant, then $x_i=\alpha(n,0,i)$ can be computed in $O(1)$ time by the formula given above. In other cases, if both i and $n-i$ are in $O(n)$, then the time costs to compute $\alpha(n,0,i)$ by the formula given above must be $O(n)$. However, we can reduce the formula further to an explicit formula to compute each of $x_i=\alpha(n,0,i), 1\le i\le n$ in $O(1)$ time.

From the construction steps of the recursive algorithm for the problem $S(n,n,r)$, the optimal move steps can be divided into three parts, the first 2 moves, recursive moves and the last 2 moves. Thereby the n steps of the optimal move sequence $x_i, 1\le i\le n$ generated by the algorithm can also be divided into thee parts accordingly. These three parts are the first part $1\le i\le 2k$, the second part $2k<i<2k+r+3$ and the third part $2k+r+3\le i\le n$, respectively, where $k=\lfloor n/4\rfloor-1$, $r=n \mod 4$.

Every move step $i, 1\le i\le n$, corresponds to a subproblem starting at cell t, and this starting cell t can also be determined by the value of i.

In the first part of the optimal move steps, the moves are generated by the first 2 moves of the algorithm. It is not difficult to see that if i is odd then $t=2(i-1)$ otherwise $t=2(i-2)$. In this case, if i is odd then $\alpha(n,0,i)=t+1=2(i-1)+1=2i-1$, otherwise, $\alpha(n,0,i)=2n-t-4=2n-2(i-2)-4=2(n-i)$.

Similarly, in the third part of the optimal move steps, the moves are generated by the last 2 moves of the algorithm. In this case, the starting cell t of the subproblem corresponding to step i can be determined by the value of $n-i$. It is not difficult to see that if $n-i$ is odd then $t=2(n-i-1)$ otherwise $t=2(n-i)$. It can also be derived by formula (1) that, if $n-i$ is odd then $\alpha(n,0,i)=2n-t-1=2n-2(n-i-1)-1=2i+1$, otherwise, $\alpha(n,0,i)=t=2(n-i)$.

For the second part of the optimal move steps, the starting cell t of the subproblem corresponding to step i is obviously $4k$. There are $2k$ steps in the first part of the optimal move steps generated before this part and thus the step i corresponds to the step $j=i-2k$ of the subproblem. It follows from $n=4\lfloor n/4\rfloor+r=4k+r+4$ that $n-t=n-4k=r+4$. It can now be derived by formula (1) that $\alpha(n,0,i)=t+d(n-t,j)=4k+d(r+4,i-2k)$.

Summing up, we have

where, $k=\lfloor n/4\rfloor-1$, $r=n \mod 4$.

Table III
SMALL TWO DIMENSIONAL ARRAY d_1

n \ i	1	2	3	4	5	6
3	1	4	0	0	0	0
4	0	3	6	0	0	0
5	2	6	1	8	0	0
6	1	7	2	5	10	0

It is obvious that the optimal move sequence $x_i, 1\le i\le n$ of the problem $S(n,n,r)$ of size n can be easily computed in optimal $O(n)$ time, since for each individual step i, its optimal move x_i can be computed in $O(1)$ time by the explicit formula (2) described above.

2) The Explicit Solutions of the Problem $S(n,n,l)$: For the general cases of the subproblems of size n, starting at cells t, if the corresponding step $j, 1\le j\le n$ of the optimal solution is denoted as $\beta(n,t,j)$, then for the problem of size n, the optimal move $x_i, 1\le i\le n$ must be $\beta(n,0,i), 1\le i\le n$. We have noticed that if $y=(y_1,\cdots,y_n)$ is an optimal solution for the problem $S(n,n,r)$, then $x=(2n-y_1,\cdots,2n-y_n)$ is also an optimal solution for the problem $S(n,n,l)$. Therefore, the two formulas (1) and (2) can be changed accordingly for the problem $S(n,n,l)$ as follows.

$$\beta(n,t,j)=\begin{cases} t+2n-d(n-t,j) & 3<n-t \text{ and } n-t<8\\ 2n-t-1 & j=1\\ t+4 & j=2\\ t+1 & j=n-t-1\\ 2n-t & j=n-t\\ \beta(n,t+4,j-2) & \text{otherwise}\end{cases} \tag{3}$$

where, $k=\lfloor n/4\rfloor-1$, $r=n \mod 4$.

3) The Explicit Solutions of the Problems $S^{-1}(n,n,r)$ and $S^{-1}(n,n,l)$: It is obvious that the explicit solution for the Problem $S^{-1}(n,n,r)$ is the formula (1) and the explicit solution for the Problem $S^{-1}(n,n,l)$ is the formula (2).

B. The Explicit Solutions of the Problems of Type $(n,n-1)$

1) The Explicit Solutions of the Problem $S(n,n-1,r)$: For the small size cases of $n=3,4,5,6$, the optimal solutions of the problem $S(n,n-1,r)$ can be listed as a small two dimensional array d_1 as follows.

For the problems of size $n=3,4,5,6$, the step $i, 1\le i\le n$ of the optimal solution can be expressed as $d_1(n,i)$. For the general cases of the subproblems of size n, starting at cells t, if the corresponding step $j, 1\le j\le n$ of the optimal solution is denoted as $\lambda_1(n,t,j)$, then for the problem of size n, the optimal move $x_i, 1\le i\le n$ must be $\lambda_1(n,0,i), 1\le i\le n$.

$$x_i = \begin{cases} 4k+d(r+4,i-2k) & 2k<i<2k+r+3 \\ 2i-1 & i \leq 2k \text{ and } i \text{ odd} \\ 2(n-i) & i \leq 2k \text{ and } i \text{ even} \\ 2i+1 & i \geq 2k+r+3 \text{ and } (n-i) \text{ odd} \\ 2(n-i) & i \geq 2k+r+3 \text{ and } (n-i) \text{ even} \end{cases} \tag{2}$$

$$x_i = \begin{cases} 4k+2n-d(r+4,i-2k) & 2k<i<2k+r+3 \\ 2(n-i)+1 & i \leq 2k \text{ and } i \text{ odd} \\ 2i & i \leq 2k \text{ and } i \text{ even} \\ 2(n-i)-1 & i \geq 2k+r+3 \text{ and } (n-i) \text{ odd} \\ 2i & i \geq 2k+r+3 \text{ and } (n-i) \text{ even} \end{cases} \tag{4}$$

Table IV
SMALL TWO DIMENSIONAL ARRAY d_2

n \ i	1	2	3	4	5	6
3	1	3	0	0	0	0
4	0	0	0	0	0	0
5	6	1	8	5	0	0
6	1	8	1	4	7	0

Table V
SMALL TWO DIMENSIONAL ARRAY d_3

n \ i	1	2	3	4	5	6
3	4	1	5	0	0	0
4	7	4	1	7	0	0
5	7	3	8	1	9	0
6	10	4	9	6	1	11

According to the algorithm presented in previous section, the function $\lambda_1(n,t,j)$ can be computed as follows.

$$\lambda_1(n,t,j) = \begin{cases} d_1(n,j) & 3 \leq n \text{ and } n \leq 6 \\ t+1 & n>6 \text{ and } j=1 \\ 2(n-t-1) & n>6 \text{ and } j=n-t-1 \\ t & n>6 \text{ and } j=n-t \\ 1+\beta(n-3,0,j-1) & \text{otherwise} \end{cases} \tag{5}$$

where $\beta(n-3,0,j-1)$ can be computed by formula (3).

If i or $n-i$ is a constant, then $x_i = \lambda_1(n,0,i)$ can be computed in $O(1)$ time by the formula given above. By using formula (4), we can reduce the formula further to an explicit formula to compute each of $x_i = \lambda_1(n,0,i), 1 \leq i \leq n$ in $O(1)$ time.

where, $k = \lfloor (n-3)/4 \rfloor - 1$, $r = (n-3) \mod 4$.

2) The Explicit Solutions of the Problem $S(n-1,n,r)$: For the small size cases of $n = 3,4,5,6$, the optimal solutions of the problem $S(n-1,n,r)$ can be listed as a small two dimensional array d_2 as follows.

For the problems of size $n = 3,4,5,6$, the step $i, 1 \leq i \leq n$ of the optimal solution can be expressed as $d_2(n,i)$. For the general cases of the subproblems of size n, starting at cells t, if the corresponding step $j, 1 \leq j \leq n$ of the optimal solution is denoted as $\lambda_2(n,t,j)$, then for the problem of size n, the optimal move $x_i, 1 \leq i \leq n$ must be $\lambda_2(n,0,i), 1 \leq i \leq n$.

According to the algorithm presented in previous section, the function $\lambda_2(n,t,j)$ can be computed as follows.

$$\lambda_2(n,t,j) = \begin{cases} d_2(n,j) & 3 \leq n \text{ and } n \leq 6 \\ n-2 & n>6 \text{ and } j=1 \\ 2n-4 & n>6 \text{ and } j=2 \\ \alpha(n-2,0,j-2) & \text{otherwise} \end{cases} \tag{7}$$

where $\alpha(n-2,0,j-2)$ can be computed by formula (1).

If i or $n-i$ is a constant, then $x_i = \lambda_2(n,0,i)$ can be computed in $O(1)$ time by the formula given above. By using formula (2), we can reduce the formula further to an explicit formula to compute each of $x_i = \lambda_2(n,0,i), 1 \leq i \leq n$ in $O(1)$ time.

where, $k = \lfloor (n-2)/4 \rfloor - 1$, $r = (n-2) \mod 4$.

3) The Explicit Solutions of the Problem $S(n-1,n,l)$: For the small size cases of $n = 3,4,5,6$, the optimal solutions of the problem $S(n-1,n,l)$ can be listed as a small two dimensional array d_3 as follows.

For the problems of size $n = 3,4,5,6$, the step $i, 1 \leq i \leq n$ of the optimal solution can be expressed as $d_3(n,i)$. For the general cases of the subproblems of size n, starting at cells t, if the corresponding step $j, 1 \leq j \leq n$ of the optimal solution is denoted as $\lambda_3(n,t,j)$, then for the problem of size n, the optimal move $x_i, 1 \leq i \leq n$ must be $\lambda_3(n,0,i), 1 \leq i \leq n$.

According to the algorithm presented in previous section, the function $\lambda_3(n,t,j)$ can be computed as follows.

$$x_i = \begin{cases} d_1(n,i) & 3 \le n \le 6 \\ 1 & n > 6 \text{ and } i = 1 \\ 2n-2 & n > 6 \text{ and } i = n-1 \\ 0 & n > 6 \text{ and } i = n \\ 4k + d(r+4, i-2k-1) + 1 & 2k+1 < i < 2k+r+4 \\ 2(n-i) - 2 & i \le 2k+1 \text{ and } i \text{ even} \\ 2i-1 & i \le 2k+1 \text{ and } i \text{ odd} \\ 2(n-i) - 4 & i \ge 2k+r+4 \text{ and } (n-i) \text{ odd} \\ 2i-1 & i \ge 2k+r+4 \text{ and } (n-i) \text{ even} \end{cases} \tag{6}$$

$$x_i = \begin{cases} d_2(n,i) & 3 \le n \le 6 \\ n-2 & n > 6 \text{ and } i = 1 \\ 2n-4 & n > 6 \text{ and } i = 2 \\ 4k + d(r+4, i-2k-2) & 2k+2 < i < 2k+r+5 \\ 2i-5 & i \le 2k+2 \text{ and } i \text{ odd} \\ 2(n-i) & i \le 2k+2 \text{ and } i \text{ even} \\ 2i-3 & i \ge 2k+r+5 \text{ and } (n-i) \text{ odd} \\ 2(n-i) & i \ge 2k+r+5 \text{ and } (n-i) \text{ even} \end{cases} \tag{8}$$

Table VI
SMALL TWO DIMENSIONAL ARRAY d_4

n \ i	1	2	3	4	5	6
3	4	2	5	0	0	0
4	0	0	0	0	0	0
5	3	8	1	4	9	0
6	10	3	0	7	4	11

Table VII
SMALL TWO DIMENSIONAL ARRAY d_5

n \ i	1	2	3	4	5	6
3	4	1	5	0	0	0
4	3	6	0	7	0	0
5	8	1	6	2	9	0
6	2	5	10	7	1	11

$$\lambda_3(n,t,j) = \begin{cases} d_3(n,j) & 3 \le n \text{ and } n \le 6 \\ 2n-2 & n > 6 \text{ and } j = 1 \\ 1 & n > 6 \text{ and } j = n-1 \\ 2n-1 & n > 6 \text{ and } j = n \\ 4 + \alpha(n-3, 0, j-1) & \text{otherwise} \end{cases} \tag{9}$$

where $\alpha(n-3, 0, j-1)$ can be computed by formula (1).

If i or $n-i$ is a constant, then $x_i = \lambda_3(n,0,i)$ can be computed in $O(1)$ time by the formula given above. By using formula (2), we can reduce the formula further to an explicit formula to compute each of $x_i = \lambda_3(n,0,i), 1 \le i \le n$ in $O(1)$ time.

where, $k = \lfloor (n-3)/4 \rfloor - 1$, $r = (n-3) \mod 4$.

4) The Explicit Solutions of the Problem $S(n, n-1, l)$: For the small size cases of $n = 3, 4, 5, 6$, the optimal solutions of the problem $S(n, n-1, l)$ can be listed as a small two dimensional array d_4 as follows.

For the problems of size $n = 3, 4, 5, 6$, the step $i, 1 \le i \le n$ of the optimal solution can be expressed as $d_4(n,i)$. For the general cases of the subproblems of size n, starting at cells t, if the corresponding step $j, 1 \le j \le n$ of the optimal solution is denoted as $\lambda_4(n,t,j)$, then for the problem of size n, the optimal move $x_i, 1 \le i \le n$ must be $\lambda_4(n,0,i), 1 \le i \le n$.

According to the algorithm presented in previous section, the function $\lambda_4(n,t,j)$ can be computed as follows.

$$\lambda_4(n,t,j) = \begin{cases} d_4(n,j) & 3 \le n \text{ and } n \le 6 \\ n+1 & n > 6 \text{ and } j = 1 \\ 3 & n > 6 \text{ and } j = 2 \\ 3 + \beta(n-2, 0, j-2) & \text{otherwise} \end{cases} \tag{11}$$

where $\beta(n-2, 0, j-2)$ can be computed by formula (3).

If i or $n-i$ is a constant, then $x_i = \lambda_4(n,0,i)$ can be computed in $O(1)$ time by the formula given above. By using formula (4), we can reduce the formula further to an explicit formula to compute each of $x_i = \lambda_4(n,0,i), 1 \le i \le n$ in $O(1)$ time.

where, $k = \lfloor (n-2)/4 \rfloor - 1$, $r = (n-2) \mod 4$.

5) The Explicit Solutions of the Problem $S^{-1}(n, n-1, r)$: For the small size cases of $n = 3, 4, 5, 6$, the optimal solutions of the problem $S^{-1}(n, n-1, r)$ can be listed as a small two dimensional array d_5 as follows.

For the problems of size $n = 3, 4, 5, 6$, the step $i, 1 \le i \le$

$$x_i = \begin{cases} d_3(n,i) & 3 \le n \le 6 \\ 2n-2 & n > 6 \text{ and } i = 1 \\ 1 & n > 6 \text{ and } i = n-1 \\ 2n-1 & n > 6 \text{ and } i = n \\ 4k + d(r+4, i-2k-1) + 4 & 2k+1 < i < 2k+r+4 \\ 2i+1 & i \le 2k+1 \text{ and } i \text{ even} \\ 2(n-i) & i \le 2k+1 \text{ and } i \text{ odd} \\ 2i+3 & i \ge 2k+r+4 \text{ and } (n-i) \text{ odd} \\ 2(n-i) & i \ge 2k+r+4 \text{ and } (n-i) \text{ even} \end{cases} \tag{10}$$

$$x_i = \begin{cases} d_4(n,i) & 3 \le n \le 6 \\ n+1 & n > 6 \text{ and } i = 1 \\ 3 & n > 6 \text{ and } i = 2 \\ 4k + d(r+4, i-2k-2) + 3 & 2k+2 < i < 2k+r+5 \\ 2(n-i)+4 & i \le 2k+2 \text{ and } i \text{ odd} \\ 2i-1 & i \le 2k+2 \text{ and } i \text{ even} \\ 2(n-i)+2 & i \ge 2k+r+5 \text{ and } (n-i) \text{ odd} \\ 2i-1 & i \ge 2k+r+5 \text{ and } (n-i) \text{ even} \end{cases} \tag{12}$$

n of the optimal solution can be expressed as $d_5(n,i)$. For the general cases of the subproblems of size n, starting at cells t, if the corresponding step $j, 1 \le j \le n$ of the optimal solution is denoted as $\lambda_5(n,t,j)$, then for the problem of size n, the optimal move $x_i, 1 \le i \le n$ must be $\lambda_5(n,0,i), 1 \le i \le n$.

According to the algorithm presented in previous section, the function $\lambda_5(n,t,j)$ can be computed as follows.

$$\lambda_5(n,t,j) = \begin{cases} d_5(n,j) & 3 \le n \text{ and } n \le 6 \\ 2n-2 & n > 6 \text{ and } j = 1 \\ 2n-5 & n > 6 \text{ and } j = 2 \\ 2n-1 & n > 6 \text{ and } j = n \\ 1 + \alpha(n-3, 0, j-2) & \text{otherwise} \end{cases} \tag{13}$$

where $\alpha(n-3,0,j-2)$ can be computed by formula (1).

If i or $n-i$ is a constant, then $x_i = \lambda_5(n,0,i)$ can be computed in $O(1)$ time by the formula given above. By using formula (2), we can reduce the formula further to an explicit formula to compute each of $x_i = \lambda_5(n,0,i), 1 \le i \le n$ in $O(1)$ time.

where, $k = \lfloor (n-3)/4 \rfloor - 1$, $r = (n-3) \mod 4$.

6) The Explicit Solutions of the Problem $S^{-1}(n-1,n,r)$: For the small size cases of $n = 3,4,5,6$, the optimal solutions of the problem $S^{-1}(n-1,n,r)$ can be listed as a small two dimensional array d_6 as follows.

Table VIII
SMALL TWO DIMENSIONAL ARRAY d_6

n \ i	1	2	3	4	5	6
3	3	1	5	0	0	0
4	0	0	0	0	0	0
5	5	8	1	6	9	0
6	5	8	1	6	4	11

For the problems of size $n = 3,4,5,6$, the step $i, 1 \le i \le n$ of the optimal solution can be expressed as $d_6(n,i)$. For the general cases of the subproblems of size n, starting at cells t, if the corresponding step $j, 1 \le j \le n$ of the optimal solution is denoted as $\lambda_6(n,t,j)$, then for the problem of size n, the optimal move $x_i, 1 \le i \le n$ must be $\lambda_6(n,0,i), 1 \le i \le n$.

According to the algorithm presented in previous section, the function $\lambda_6(n,t,j)$ can be computed as follows.

$$\lambda_6(n,t,j) = \begin{cases} d_6(n,j) & 3 \le n \text{ and } n \le 6 \\ n-2 & n > 6 \text{ and } j = n-1 \\ 2n-1 & n > 6 \text{ and } j = n \\ \beta(n-2,0,j) & \text{otherwise} \end{cases} \tag{15}$$

where $\beta(n-2,0,j)$ can be computed by formula (3).

If i or $n-i$ is a constant, then $x_i = \lambda_6(n,0,i)$ can be computed in $O(1)$ time by the formula given above. By using formula (4), we can reduce the formula further to an explicit formula to compute each of $x_i = \lambda_6(n,0,i), 1 \le i \le n$ in $O(1)$ time.

where, $k = \lfloor (n-2)/4 \rfloor - 1$, $r = (n-2) \mod 4$.

7) The Explicit Solutions of the Problem $S^{-1}(n-1,n,l)$: For the small size cases of $n = 3,4,5,6$, the optimal solutions of the problem $S^{-1}(n-1,n,l)$ can be listed as a small two dimensional array d_7 as follows.

For the problems of size $n = 3,4,5,6$, the step $i, 1 \le i \le n$ of the optimal solution can be expressed as $d_7(n,i)$. For the

$$x_i = \begin{cases} d_5(n,i) & 3 \le n \le 6 \\ 2n-2 & n > 6 \text{ and } i = 1 \\ 2n-5 & n > 6 \text{ and } i = 2 \\ 2n-1 & n > 6 \text{ and } i = n \\ 4k + d(r+4, i-2k-2) + 1 & 2k+2 < i < 2k+r+5 \\ 2i-4 & i \le 2k+2 \text{ and } i \text{ odd} \\ 2(n-i)-1 & i \le 2k+2 \text{ and } i \text{ even} \\ 2i-2 & i \ge 2k+r+5 \text{ and } (n-i) \text{ even} \\ 2(n-i)-1 & i \ge 2k+r+5 \text{ and } (n-i) \text{ odd} \end{cases} \tag{14}$$

$$x_i = \begin{cases} d_6(n,i) & 3 \le n \le 6 \\ n-2 & n > 6 \text{ and } i = n-1 \\ 2n-1 & n > 6 \text{ and } i = n \\ 4k + d(r+4, i-2k) & 2k < i < 2k+r+3 \\ 2(n-i)-3 & i \le 2k \text{ and } i \text{ odd} \\ 2i & i \le 2k \text{ and } i \text{ even} \\ 2(n-i)-5 & i \ge 2k+r+3 \text{ and } (n-i) \text{ odd} \\ 2i & i \ge 2k+r+3 \text{ and } (n-i) \text{ even} \end{cases} \tag{16}$$

Table IX
SMALL TWO DIMENSIONAL ARRAY d_7

n \ i	1	2	3	4	5	6
3	1	4	0	0	0	0
4	1	4	7	0	0	0
5	1	8	3	7	0	0
6	1	6	9	4	10	0

Table X
SMALL TWO DIMENSIONAL ARRAY d_8

n \ i	1	2	3	4	5	6
3	2	4	0	0	0	0
4	0	0	0	0	0	0
5	4	1	8	3	0	0
6	2	5	10	1	7	0

general cases of the subproblems of size n, starting at cells t, if the corresponding step $j, 1 \le j \le n$ of the optimal solution is denoted as $\lambda_7(n,t,j)$, then for the problem of size n, the optimal move $x_i, 1 \le i \le n$ must be $\lambda_7(n,0,i), 1 \le i \le n$.

According to the algorithm presented in previous section, the function $\lambda_7(n,t,j)$ can be computed as follows.

$$\lambda_7(n,t,j) = \begin{cases} d_7(n,j) & 3 \le n \text{ and } n \le 6 \\ 1 & n > 6 \text{ and } j = 1 \\ 4 & n > 6 \text{ and } j = 2 \\ 0 & n > 6 \text{ and } j = n \\ \beta(n-3,0,j-2) & \text{otherwise} \end{cases} \tag{17}$$

where $\beta(n-3,0,j-2)$ can be computed by formula (3).

If i or $n-i$ is a constant, then $x_i = \lambda_7(n,0,i)$ can be computed in $O(1)$ time by the formula given above. By using formula (4), we can reduce the formula further to an explicit formula to compute each of $x_i = \lambda_7(n,0,i), 1 \le i \le n$ in $O(1)$ time.

where, $k = \lfloor (n-3)/4 \rfloor - 1$, $r = (n-3) \mod 4$.

8) The Explicit Solutions of the Problem $S^{-1}(n, n-1, l)$: For the small size cases of $n = 3, 4, 5, 6$, the optimal solutions of the problem $S^{-1}(n, n-1, l)$ can be listed as a small two dimensional array d_8 as follows.

For the problems of size $n = 3, 4, 5, 6$, the step $i, 1 \le i \le n$ of the optimal solution can be expressed as $d_8(n,i)$. For the general cases of the subproblems of size n, starting at cells t, if the corresponding step $j, 1 \le j \le n$ of the optimal solution is denoted as $\lambda_8(n,t,j)$, then for the problem of size n, the optimal move $x_i, 1 \le i \le n$ must be $\lambda_8(n,0,i), 1 \le i \le n$.

According to the algorithm presented in previous section, the function $\lambda_8(n,t,j)$ can be computed as follows.

$$\lambda_8(n,t,j) = \begin{cases} d_8(n,j) & 3 \le n \text{ and } n \le 6 \\ n+1 & n > 6 \text{ and } j = n-1 \\ 0 & n > 6 \text{ and } j = n \\ 3 + \alpha(n-2,0,j) & \text{otherwise} \end{cases} \tag{19}$$

where $\alpha(n-2,0,j)$ can be computed by formula (1).

If i or $n-i$ is a constant, then $x_i = \lambda_8(n,0,i)$ can be computed in $O(1)$ time by the formula given above. By using formula (2), we can reduce the formula further to an explicit formula to compute each of $x_i = \lambda_8(n,0,i), 1 \le i \le n$ in $O(1)$ time.

$$x_i = \begin{cases} d_7(n,i) & 3 \le n \le 6 \\ 1 & n > 6 \text{ and } i = 1 \\ 4 & n > 6 \text{ and } i = 2 \\ 0 & n > 6 \text{ and } i = n \\ 4k + d(r+4, i-2k-2) + 4 & 2k+2 < i < 2k+r+5 \\ 2(n-i)+1 & i \le 2k+2 \text{ and } i \text{ odd} \\ 2i & i \le 2k+2 \text{ and } i \text{ even} \\ 2(n-i)-1 & i \ge 2k+r+5 \text{ and } (n-i) \text{ even} \\ 2i & i \ge 2k+r+5 \text{ and } (n-i) \text{ odd} \end{cases} \tag{18}$$

$$x_i = \begin{cases} d_8(n,i) & 3 \le n \le 6 \\ n+1 & n > 6 \text{ and } i = n-1 \\ 0 & n > 6 \text{ and } i = n \\ 4k + d(r+4, i-2k) + 3 & 2k < i < 2k+r+3 \\ 2i+2 & i \le 2k \text{ and } i \text{ odd} \\ 2(n-i)-1 & i \le 2k \text{ and } i \text{ even} \\ 2i+4 & i \ge 2k+r+3 \text{ and } (n-i) \text{ odd} \\ 2(n-i)-1 & i \ge 2k+r+3 \text{ and } (n-i) \text{ even} \end{cases} \tag{20}$$

where, $k = \lfloor (n-2)/4 \rfloor - 1$, $r = (n-2) \mod 4$.

V. Concluding Remarks

We have studied generalized shuffling problems of size n. It has been proved that the minimum number of moves needed to play the game of size n is n. In the section 3, the optimal recursive construction algorithms which can produce an optimal solution in n moves for very large size n is presented. Finally, in the section 4, the extremely simple explicit solutions for the optimal moving sequences of the 12 different generalized shuffling problems of size n is given. The formula gives for each individual step i, its optimal move in $O(1)$ time.

For the interesting generalized shuffling problems, some research problems are open. If some constraints are added to the move sequences, then the problem become complicated. For example, we can add a restriction to a move that only two adjacent coins of different colors can be moved. For this constraint shuffling problem, what is its optimal solution? Can we find an efficient algorithm to generate optimal solutions for the constraint shuffling problem in the time proportional to the output size? This is also an open problem. We will investigate the these problems further.

References

[1] R. Bird, Pearls of Functional Algorithm Design, 258-274, Cambridge University Press, 2010.

[2] Erik D. Demaine, Playing games with algorithms, Algorithmic combinatorial game theory. Proceedings of the 26th Symposium on Mathematical Foundations in Computer Science, LNCS 2136, 18-32, 2001.

[3] Erik D. Demaine and Martin L. Demaine, Puzzles, Art, and Magic with Algorithms, Theory of Computing Systems, vol. 39, number 3, 473-481, 2006.

[4] A. Levitin and M. Levitin, Algorithmic Puzzles, 3-31, Oxford University Press, New York, 2011.

[5] J. Kleinberg, E. Tardos. Algorithm Design, 223-238, Addison Wesley, 2005.

[6] D.L. Kreher and D. Stinson, Combinatorial Algorithms: Generation, Enumeration and Search, 125-133, CRC Press, 1998.

[7] Hiromi Miyajima, Masataka Fujisaki, and Noritaka Shigei, Quantum Search Algorithms in Analog and Digital Models,IAENG International Journal of Computer Science, Vol. 39, No. 2, 182-189, 2012.

[8] D. Zhu, L. Wang, J. Tian and X. Wang, An Algorithmic Solution for a Single Player Computer Game, International Journal of Applied Mathematics & Statistics, Vol. 52, No. 5, 21-28, 2014.

[9] D. Zhu, Y. Wu, L. WangX. Wang, Complete Solutions for a Combinatorial Puzzle in Linear Time and Its Computer Implementation, Appl. Math. Inf. Sci. 9:2, 1-14, 2015.