Modern Applied Science
www.ccsenet.org/mas
Vol. 4, No. 1
January 2010

Chessboard Coverage Teaching Based on Divide-and-Conquer Algorithm

Zhijie Li
Institute of Nonlinear Information Technology, Dalian Nationalities University
Dalian 116600, China
E-mail: lizhijie@dlnu.edu.cn

Feixue Huang
Department of Economics, Dalian University of Technology
Dalian 116024, China

Xiangdong Liu & Xiaodong Duan
Institute of Nonlinear Information Technology, Dalian Nationalities University
Dalian 116600, China

Abstract

Though the divide-and-conquer algorithm is a powerful technology, but it is difficult to be used in practice, so the design of this algorithm should be regulated in the teaching. In this article, the teaching process and the characters of the divide-and-conquer algorithm are studied when solving the problem of chessboard coverage. Aiming at the deficiency that the existing teaching method uses skills in decomposing sub-problems and increases the teaching difficulty, the coverage sequence of L-type dominoes is improved. The improved algorithm can keep consistent with the divide-and conquer strategy and standardize the iterative process, and increase the normative character and the consistence of the algorithm, and hence achieve ideal teaching result.

Keywords: Algorithm, Divide-and-conquer, Chessboard coverage

1. Introduction

Algorithm design and analysis (Cormen TH, 2001 & Wang, 2005, P.124-127 & Goodrich MT, 2007 & Wang, 2007) is the comprehensive lesson integrating the application, creation, and practice. To solve problem by the algorithm, the practice problem should be firstly abstracted as the mathematical model, and then the algorithm corresponding to this model is designed. These approaches are an organic integer, and the algorithm design is the core content in them, the key approach from proposing problem to solving problem. In the teaching process, proper method should be adopted to make students grasp the basic theory and technology of the algorithm. The standardization of the algorithm design is the important factor to influence students' understandings about the algorithm ideas. This article mainly studies the algorithm design of the divide-and-conquer algorithm solving the problem of chessboard coverage, and explores the standardization of the algorithm design.

The computation time needed by any one problem which can be solved by computer is related with the scale of the problem. The scale of the problem is smaller, the needed computation time is always less, and the computation is easier. The deign idea of the divide-and-conquer algorithm (Stephen RM, 2004, P. 420-425 & Shimojoa F, 2005, 151-164 & Min F, 2005, P. 1277-1286 & Huang, 2008, 1831-1839 & Wu YZ, 2008, P. 41-44) is to divide a complex big problem into many small problems and conquer them one by one. The concrete approaches are to divide the big problem which is difficult to be solved directly into two or more same or similar sub-problems, and divide the sub-problems into smaller sub-problems until these sub-problems can be directly solved simply, and the solution of the original problem is the combination of the solutions of all sub-problems. One main implementation mode of this approach in super language is the recursion. Therefore, the divide-and-conquer is a kind of important algorithm, and its approaches seem simply, but contain profound philosophical ideas, and this skill is the base of many high-efficiency algorithms.

One representative application of the divide-and-conquer is to solve the chessboard coverage problem (Ohio C, 2007, P.464-469). In the teaching process nowadays, the solving of the algorithm is excessively to pursue the skill, which induces that the algorithm flow is not consistent with the divide-and-conquer idea. Though the correctness of the algorithm is not influenced, but it brings negative influences for students to understand the problem and the divide-and-conquer algorithm. Therefore, the process of the divide-and-conquer algorithm to solving the chessboard coverage problem is mainly discussed in the article, and the standardization and consistence of the solving algorithm are improved for obtaining better teaching effect.

2. Divide-and-conquer algorithm

2.1 Basic ideas

For the problem with the scale of n, if n is small, the problem can be solved directly, or else, the problem can be divided into *k* sub-problems with small scale, and $1<k\leq n$, and these sub-problems all can be solved, and are independent each other and have same form with the original problem. Solve the sub-problems recursively, and combine the solutions of various sub-problems, so the solution of the original problem is obtained. This strategy of algorithm design is called as the divide-and-conquer method. The sub-problem generated in the divide-and-conquer algorithm is always the smaller mode of the original problem, which offers the convenience to use the recursion technology. Under this situation, by repetitively using the measure of divide-and-conquer, the type of the sub-problems can be consistent with the original problem, but the scale continually reduced until the sub-problem can be easily and directly solved. So the recursion process is generated naturally. The application of the divide-and-conquer algorithm and the recursion in the algorithm design can generate higher efficiency. The problems which can be solved by the divide-and-conquer algorithm generally have following characters.

(1) When the scale of the problem is reduced to certain extent, it can be easily solved.

(2) The problem can be divided into many same problems with small scale, i.e. the problem has the character of the optimal sub-structure.

(3) The solutions of the sub-problems of the problem can be combined as the solution of the problem.

(4) Various sub-problems divided by this problem are independent each other, i.e. the public problem is not contained among sub-problems.

2.2 Solving approaches

On each level of recursion, the divide-and-conquer algorithm has three approaches. First, the original problem is divided into many sub-problems with small scale which are independent each other, and have same form with the original problem. Second, if these sub-problems have small scale and can be easily solved, directly solve them, or else, recursively solve various sub-problems. Finally, combine the solutions of various sub-problems as the solution of the original problem. The general algorithm design mode can be described as follows.

```
divide-and-conquer(P)
{
   if (|P|<=n0) adhoc(P);
   divide P into smaller subinstances P1 ,P2 ,...,Pk;
   for (i=1; i<=k; i++)
      yi = divide-and-conquer(Pi);
   return merge(y1,y2,...,yk)
}
```

Where, |P| denotes the scale of the problem P, n_0 is one threshold value which denotes that the problem can be solved easily and directly and the problem needs not be further divided when the scale of the problem P doesn't exceed n_0, adhoc (P) is the basic sub-algorithm in the divide-and-conquer algorithm which can be used to directly solve the problem P with small scale. So when the scale of P doesn't exceed n_0, it can be directly solved by the algorithm adhoc (P). The algorithm of merge ($y_1,y_2,...,y_k$) is the combining sub-algorithm in the divide-and-conquer algorithm which is used to combine the corresponding solutions $y_1,y_2,...,y_k$ of the sub-problems P_1 ,P_2 ,...,P_k into the solution of P. According to the division principle of the divide-and-conquer algorithm, when designing the algorithm by the divide-and-conquer algorithm, the scales of sub-problems should be approximately same. In another word, the processing method dividing one problem into *k* sub-problems with same scale is feasible. This method is based on the balance of sub-problems.

3. Divide-and-conquer strategy of the chessboard coverage problem

3.1 Problem denotation in teaching

In the chessboard composed by $2^k \times 2^k$ panes, if there is one pane which is different with other panes, this pane is called as the special pane, and this chessboard is called as the special chessboard. Obviously, the positions that the special pane occurs in the chessboard include 4^k sorts. For any $k \geq 0$, there are 4^k sorts of different special chessboard. The special chessboard in Figure 1 is one of 16 special chessboard when $k = 2$.

In the problem of chessboard coverage, 4 kinds of L-type dominos with different forms seen in Figure 2 should cover all panes except for special panes in the appointed special chessboard, and any two L-type dominos can not be covered repetitively. Obviously, in any one chessboard coverage of $2^k \times 2^k$, the amount of used L-type dominos is just $(4^k-1)/3$.

A simple algorithm about the chessboard coverage problem can be designed by the divide-and-conquer strategy. When $k > 0$, divide the chessboard of $2^k \times 2^k$ into 4 sub-chessboards of $2^{k-1} \times 2^{k-1}$ (seen in Figure 3(a)). The special pane must be located in the small one of four sub-chessboards, and other three sub-chessboards have no special panes. To convert these three sub-chessboards into the special chessboard, one L-type domino can be put on the junction of three smaller chessboards (seen in Figure 3(b)). The pane which is covered by the L-type domino on three sub-chessboards is the special pane on this chessboard, so the original problem is converted into four smaller chessboard coverage problems. Recursively use this division until the chessboard is simplified as the chessboard of 1×1.

Taking the special chessboard in Figure 1 as the example, the final chessboard coverage effect is seen in Figure 4. In generally teaching, to lively play the result of chessboard coverage, different colors are generally adopted to distinguish neighboring L-type dominos. However, as viewed from the actual teaching effect, the colorful pane will easily generate confusion of vision, and can not make students to clearly understand the coverage result. And the demonstration program of the colorful chessboard coverage is relatively complex. These factors show that the denotation of the chessboard coverage needs to be further studied to better adapt the teaching demands.

3.2 Problems existing in the algorithm

The total idea of chessboard coverage algorithm based on the divide-and-conquer strategy in the teaching materials is to divide the chessboard into four sub-chessboards first, and then respectively cover these four sub-chessboards. For each sub-chessboard, the program will first judge whether the special pane is in this sub-chessboard. If it is in the sub-chessboard, the program will recursively transfer the program to cover this sub-chessboard, or else, cover the neighboring panes with other three sub-chessboards, and then recursively transfer the program to cover this sub-chessboard. The concrete chessboard coverage algorithm ChessBoard can be described as follows.

```
void ChessBoard(int tr, int tc, int dr, int dc, int size)
{
if(size==1) return;
int t=tile++, s=size/2;
// cover the sub-chessboard of the top left corner
if(dr<tr+s && dc<tc+s)
ChessBoard(tr, tc, dr, dc, s);
else{Board[tr+s-1][tc+s-1]=t;
ChessBoard(tr, tc, tr+s-1, tc+s-1, s); }
// cover the sub-chessboard of the top right corner
if(dr<tr+s && dc>=tc+s)
ChessBoard(tr, tc+s, dr, dc, s);
else{Board[tr+s-1][tc+s]=t;
ChessBoard(tr, tc+s, tr+s-1, tc+s, s); }
// cover the sub-chessboard of the down left corner
if(dr>=tr+s && dc<tc+s)
ChessBoard(tr+s, tc, dr, dc, s);
else{Board[tr+s][tc+s-1]=t;
ChessBoard(tr+s, tc, tr+s, tc+s-1, s); }
// cover the sub-chessboard of the down right corner
```

```
if(dr>=tr+s && dc>=tc+s)
ChessBoard(tr+s, tc+s, dr, dc, s);
else{Board[tr+s][tc+s]=t;
ChessBoard(tr+s, tc+s, tr+s, tc+s, s); }
}
```

In above algorithm, a two-dimensional integer array Board is used to denote the chessboard. Board[0][0] is the pane of top left corner of the chessboard. Tile is a comprehensive integer variable in the algorithm and it is used to denote the number of the L-type domino, and its initial value is 0. And the input parameters of the algorithm include tr (the row number of the pane of top left corner of the chessboard), tc (the column number of the pane of top left corner of the chessboard), dr (the row number of the special pane), dc (the column number of the special pane), size (size=2^k, and it denotes that the specification of chessboard is $2^k \times 2^k$). Taking the special chessboard in Figure 1 as the example, according to the algorithm of ChessBoard, the sequence of chessboard coverage is seen in Figure 5, where, each three same numbers denotes the L-type domino of one type.

From the coverage sequence of Figure 5, though the ChessBoard algorithm uses the skill of algorithm design, but its implementation process is not completely consistent with the divide-and-conquer strategy. According to the divide-and-conquer strategy, after the algorithm judge the position of the special pane, it should use the L-type domino to cover the junction of three sub-chessboards. But the algorithm of ChessBoard splits this approach, i.e. the L-type domino at the junction of three sub-chessboards is divided into three parts which are implemented respectively in different program blocks. Because the algorithm adopts the recursive transfer, the coverage process of each sub-chessboard should follow this rule. Though the implementation process of the ChessBoard algorithm is correct, but it doesn't accord with the above divide-and-conquer strategy which uses one L-type domino to cover, so students will feel difficult to understand the problem and accept the idea of the divide-and-conquer algorithm. Because the divide-and-conquer is difficult, so if the design skill is added in the implementation of the algorithm, students will be afraid of difficult to study this algorithm and good teaching effect will not be realized.

4. Optimization of teaching contents

4.1 Optimization of problem denotation

First, aiming at the problem in the section of 3.1, optimize the problem denotation of chessboard coverage. Because in the teaching process, the colorful pane is easily confused in vision, and it makes against the understanding of the problem, so the denotation of the chessboard coverage is improved in the article, which can help students to totally grasp the problem as a whole, and clearly know the coverage result. Because the L-type domino is used to cover, so the polygonal lines with different directions can respectively denote four sorts of L-type dominos (seen in Figure 6). Comparing with the colorful coverage result in Figure 4, the result of Figure 6 is more simple and clear, and it is easy to be used in the teaching process.

4.2 Optimization of the solving algorithm

Next, optimize the solving algorithm. Though the ChessBoard algorithm is correct, but it makes against the teaching of the divide-and-conquer algorithm because it uses the skill of program design. To implement the standardization of the teaching content and reduce the difficulty of solving problem, the algorithm skill needs to be eliminated. The following algorithm ChessBoard 1 improves the ChessBoard algorithm, standardizes the flow of the algorithm, and makes it to be consistent with the divide-and-conquer strategy. Its total idea is to divide the chessboard into four sub-chessboards, and especially judge whether the special pane is in these four sub-chessboards. If it is in the sub-chessboard, use the L-type domino to cover the junction of other three sub-chessboards. When the judgment is over, transfer the program to cover these four sub-chessboards one by one. The concrete algorithm can be described as follows.

```
void ChessBoard1(int tr, int tc, int dr, int dc, int size)
{
if(size==1) return;
int t=tile++, s=size/2;
// Use (d) type domino to cover the junction of the chessboards
if(dr<tr+s && dc<tc+s)
{ Board[tr+s-1][tc+s]=t; Board[tr+s][tc+s-1]=t; Board[tr+s][tc+s]=t;
int dr1=dr, dc1=dc, dr2= tr+s-1, dc2= tc+s, dr3= tr+s, dc3= tc+s-1, dr4= tr+s, dc4= tc+s;}
// Use (c) type domino to cover the junction of the chessboards
```

```
if(dr<tr+s && dc>=tc+s)
{ Board[tr+s-1][tc+s-1]=t; Board[tr+s][tc+s-1]=t; Board[tr+s][tc+s]=t;
int dr1=tr+s-1, dc1=tc+s-1, dr2= dr, dc2= dc, dr3= tr+s, dc3= tc+s-1, dr4= tr+s, dc4= tc+s;}
// Use (b) type domino to cover the junction of the chessboards
if(dr>=tr+s && dc<tc+s)
{ Board[tr+s-1][tc+s-1]=t; Board[tr+s-1][tc+s]=t; Board[tr+s][tc+s]=t;
int dr1=tr+s-1, dc1=tc+s-1, dr2= tr+s-1, dc2=tc+s, dr3=dr, dc3=dc, dr4= tr+s, dc4= tc+s;}
// Use (a) type domino to cover the junction of the chessboards
if(dr>=tr+s && dc>=tc+s)
{ Board[tr+s-1][tc+s-1]=t; Board[tr+s-1][tc+s]=t; Board[tr+s][tc+s-1]=t;
int dr1=tr+s-1, dc1=tc+s-1, dr2=tr+s-1, dc2=tc+s, dr3=tr+s, dc3=tc+s-1, dr4=dr, dc4=dc;}
// Respectively cover four sub-chessboards
ChessBoard1(tr, tc, dr1, dc1, s);
ChessBoard1(tr, tc+s, dr2, dc2, s);
ChessBoard1(tr+s, tc, dr3, dc3, s);
ChessBoard1(tr+s, tc+s, dr4, dc4, s);
}
```

In above algorithm, the meanings of the two-dimensional array Board, the number tile of the L-type domino, and the input parameters tr, tc, dr, dc, and size are same with the algorithm of ChessBoard1. The difference is that after the L-type domino covers the junction of four sub-chessboards, the program redefines the row number and column number of four special panes for these four sub-chessboards, i.e. redefining 8 local integer variables: the row number dr1 and the column number dc1 of the special pane in the first sub-chessboard, the row number dr2 and the column number dc2 of the special pane in the second sub-chessboard, the row number dr3 and the column number dc3 of the special pane in the third sub-chessboard, the row number dr4 and the list number dc4 of the special pane in the fourth sub-chessboard. In another word, after L-type domino covers the junction of sub-chessboards, four sub-chessboards all change as the special sub-chessboards with their own special pane. Therefore, the special panes of various sub-chessboards must be redefined, and 8 local integer variables are used to transfer the row numbers and column numbers of the special panes in these four special sub-chessboards. To compare with the ChessBoard algorithm, taking the special chessboard in Figure 1 as the example, according to the algorithm of ChessBoard1, the sequence of the chessboard coverage is seen in Figure 7.

In the comparison, the chessboard coverage process of the ChessBoard1 algorithm is consistent with the solving approach with the divide-and-conquer strategy. Different with the ChessBoard algorithm which splits the L-type domino to cover, the ChessBoard1 algorithm is transferred recursively once, and generates a L-type domino, which simplifies the implementation of the divide-and-conquer in the algorithm teaching, and liberates students from complex concrete computation, so students' learning target will be more specific, and they will mainly understand the basic idea and solving approaches of the divide-and-conquer algorithm, and grasp the essential of the problem, and need not waste too much energy in the skill of program design.

5. Analysis of algorithm complexity

From the general design mode of the divide-and-conquer algorithm, the algorithm is a recursive process. Therefore, the computation efficiency of the divide-and-conquer algorithm can be analyzed by the recursion equation. For the convenience, suppose that the problem which division threshold value n_0=1 and the solution scale of the adhoc algorithm is 1 consumes 1 unit time. And suppose that the divide-and-conquer algorithm divides the problem with the scale of n into k sub-problems with the scale of n/m, and to divide the original problem into k sub-problems and combine the solutions of k sub-problems as the solution of the original problem by the algorithm merge needs f(n) units time. If T(n) denotes the computation time needing by the problem P with the solution scale of |P|=n by the divide-and-conquer algorithm, so

$$\begin{cases} T(1) = 1 \\ T(n) = kT(n/m) + f(n) \end{cases} \tag{1}$$

By the iterative method of solving the recursion equation introduced in the asymptotic order solution in the complexity

of the algorithm, the solution of the equation (1) can be solved.

$$T(n) = n^{\log_m k} + \sum_{j=0}^{\log_m n-1} k^j f(n/m^j) \tag{2}$$

In the algorithm of ChessBoard of the chessboard coverage problem, suppose that T(k) is the time that the algorithm of ChessBoard1 covers one chessboard of $2^k \times 2^k$, so from the divide-and-conquer strategy equation (1) of the algorithm, T(k) satisfies following recursion equation.

$$T(k) = \begin{cases} O(1) & k = 0 \\ 4T(k\text{-}1) + O(1) & k > 0 \end{cases} \tag{3}$$

According to the equation (2), solve this recursion equation, and obtain T(k) = O(4^k). Because the amount of L-type domino to cover one chessboard of $2^k \times 2^k$ is (4^k-1)/3, the algorithm of ChessBoard1 is the optimal algorithm on the asymptotic meaning.

6. Conclusions

In general algorithm strategy, the divide-and-conquer algorithm is one of the most important sorts, and its establishment principle is to divide and rule, so it is the base of many high-efficiency algorithms. The application of the divide-and-conquer algorithm in the solving of chessboard coverage problem is studied in the article. Three approaches of the divide-and-conquer algorithm to solve the chessboard coverage problem are an organic integer, and the division of the original problem into many sub-problems is the key content, and it is the key approach to solve problem by the divide-and-conquer algorithm. The present divide-and-conquer method uses program skill to solve the problem of chessboard coverage, and splits the L-type dominos and cover, which is not consistent with the former analysis, and adds difficulties for the divide-and-conquer algorithm which is not complex originally, and the teaching effect is not ideal. The divide-and-conquer method contains abundant mathematical ideas which should be dug and studied in the teaching to ensure the continuity and consistence of students' thinking development and enhance the learning effect. Therefore, aiming at the problem of chessboard coverage, the coverage sequence of L-type dominos in the flow of the divide-and-conquer algorithm is improved, and the improved algorithm can not only eliminate the algorithm skill, but add the standardization and consistence of the algorithm and achieve better teaching effect.

References

Cormen TH, Leiserson CE, Rivest RL, et al. (2001). *Introduction to Algorithms (Second Edition).* MIT Press. 2001.

Goodrich MT & Tamassia R. (2007). *Algorithm Analysis and Design.* Beijing: People's Posts & Telecom Press.

Huang, Jing, Liu, Dayou & Yang, Bo, et al. (2008). A Self-organization Based Divide and Conquer Algorithm for Distributed Constraint Optimization Problems. *Journal of Computer Research and Development.* No. 45(11). P. 1831-1839.

Min F, Xie LJ & Liu QH. (2005). A Divide-and-Conquer Discretization Algorithm. *Lecture Notes in Computer Science.* No. 3613. P. 1277-1286.

Ohio C. (2007). Learning: An Effective Approach in Endgame Chess Board Evaluation. *Proceedings of the Sixth International Conference on Machine Learning and Applications.* P.464-469.

Shimojoa F, Kalia RK & Nakano A, et al. (2005). Embedded divide-and-conquer algorithm on hierarchical real-space grids: parallel molecular dynamics simulation based on linear-scaling density functional theory. *Computer Physics Communications.* No. 167(3). P. 151-164.

Stephen RM, Brian JS & Karen W, et al. (2004). Divide and Conquer. *Psychological Science.* No. 15(6). P. 420-425.

Wang, Suli & Bai, Shouhua. (2005). Teaching Method of Algorithm Analysis and Design. *Journal of Xiangtan Normal University (Natural Science Edition).* No. 27(3). P.124-127.

Wang, Xiaodong. (2007). *Computer Algorithm Design and Analysis.* Beijing: Electronics Industry Publishing House.

Wu YZ, Jiang ZS & Ding JW, et al. (2008). Subdivision solving of multi-domain simulation models. *Journal of Huazhong University of Science and Technology.* No. 36(10). P. 41-44.

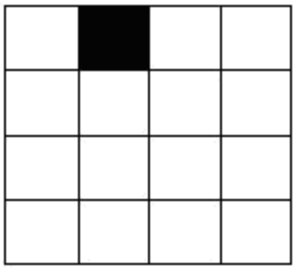

Figure 1. A Special Chessboard when k=2

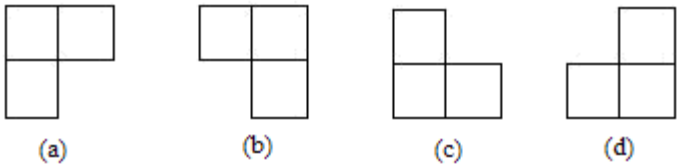

Figure 2. Four L-type Dominos with Different Forms

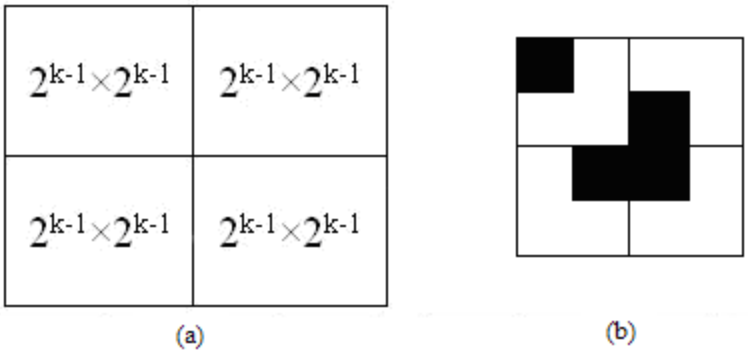

Figure 3. Division of Chessboard

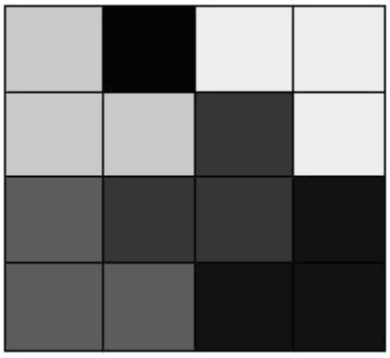

Figure 4. Result of Chessboard Coverage

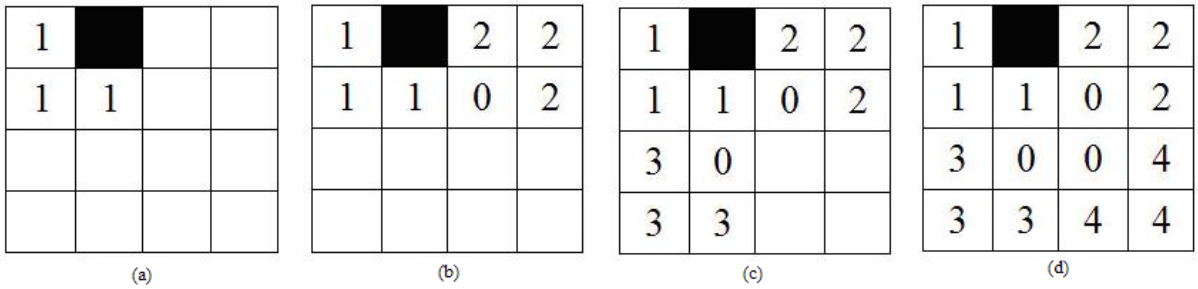

Figure 5. Chessboard Coverage Sequence of the ChessBoard Algorithm

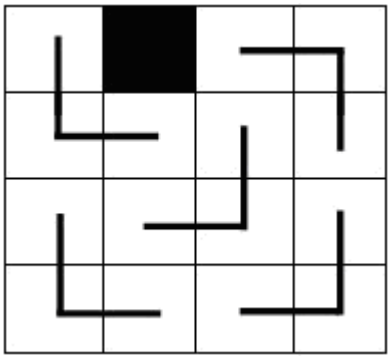

Figure 6. Simple Denotation of Chessboard Coverage Result

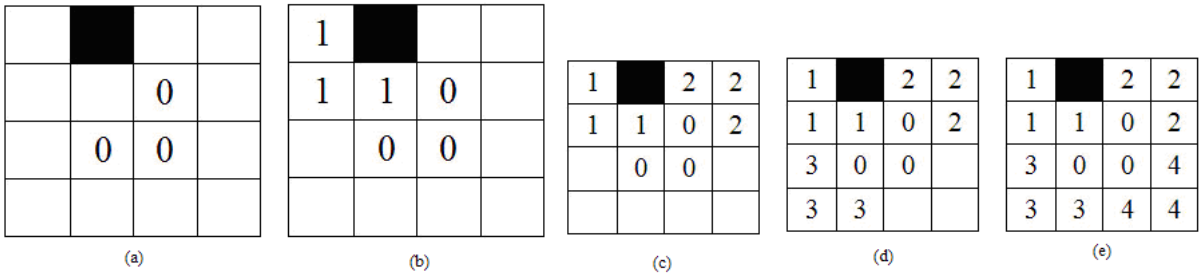

Figure 7. Chessboard Coverage Sequence of the ChessBoard1 Algorithm