Solve the Problem of Chess Board in the Shape of the Letter L with Three different Patterns Using Artificial Intelligence & Algorithms

Israa Shakir Seger
College of Basic Education, University of Muthanna, Muthanna, Iraq.

Israa M. Hayder
Department of Computer Systems Techniques, Qurna Technique Institute, STU, Basrah, Iraq.

Hussain A. Younis*
College of Education for Women, University of Basrah, Basrah, Iraq.
School of Computer Sciences, Universiti Sains Malaysia, USM, Penang, Malaysia.
E-mail: hussain.younis@uobasrah.edu.iq

Hameed Abdul-Kareem Younis
College of Computer Science and Information Technology, University of Basrah, Basrah, Iraq.

Received July 10, 2021; Accepted November 12, 2021
ISSN: 1735-188X
DOI: 10.14704/WEB/V19I1/WEB19006

Abstract

In recent years, the chess game has begun to develop successful programming solutions. Computers were programmed to play chess in the middle of the twentieth century. Computer skills have become better and higher than the skills of chess players in the world, and from here this study has made it possible to find the optimal solution for the four square pieces in the form of a letter (L) without repetition and quick access to fill the sites and voids and to complete the entire area. It is our task to cover a (2n×2n) Chessboard with L-shaped tiles each tile is a (2×2) square with a (1×1) square removed from one corner. We are working to cover the Chessboard in such a way that there is a single 1×1 box left in the 'corner' of the Chessboard (by the 'corner' we mean one corner of the box should be uncovered). In this task, we will solve this problem with three approaches, the C programming approach, the second by dividing and conquering approach and the last by a greedy method approach. Three algorithms were used and a comparison was made between them, and the fastest method was achieved by a greedy method, with eight cases comparing one and four cases, respectively.

Keywords

Dynamic Programming Approach, Divide & conquer approach, Greedy Method Approach.

Introduction

A lot of puzzles are used on the chessboard, and the most common puzzles are the wheat problem and the chessboard, the Queen's problem, the Knight's tour, the Defective Chessboard Problem (Anany Levitin and Maria Levitin, 2011). In this paper, we will talk about the defective chessboard problem. We have a 2k*2k board and we have L-shaped tiles, the task is to cover all parts of the board using this tile, except one box called Defective. In this task, you can rotate the tiles with four different orientations; Figure (1) below shows the shape you can use.

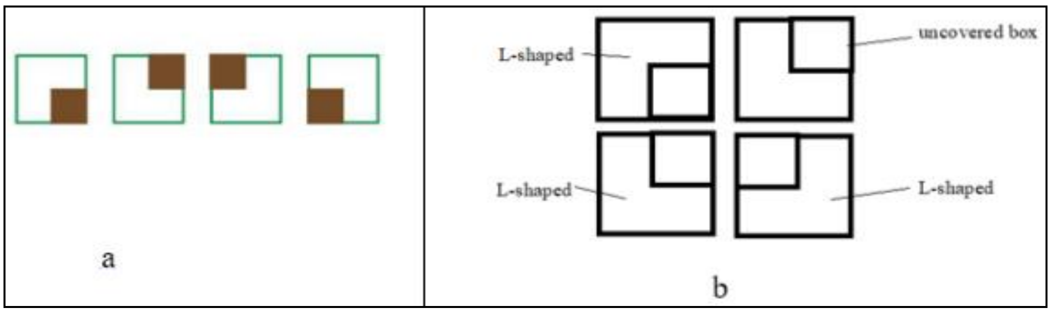


Figure 1 a. Four different tiling's of a 2*2 courtyard b. The covers L-shaped

Let k be a natural number, assuming that Claim (k) is true, i.e. assuming that a 2k*2k courtyard can be tiled with L-shaped tiles in such a way that there is a single 1 * 1 hole left anywhere us like in the courtyard. And one of the most important ways to solve this problem is by using David and Conker and that all solutions depend on his put way tile. Figure (2) below shows that, and in this paper, we will use shape (C) in Figure 2 to solve the problem.

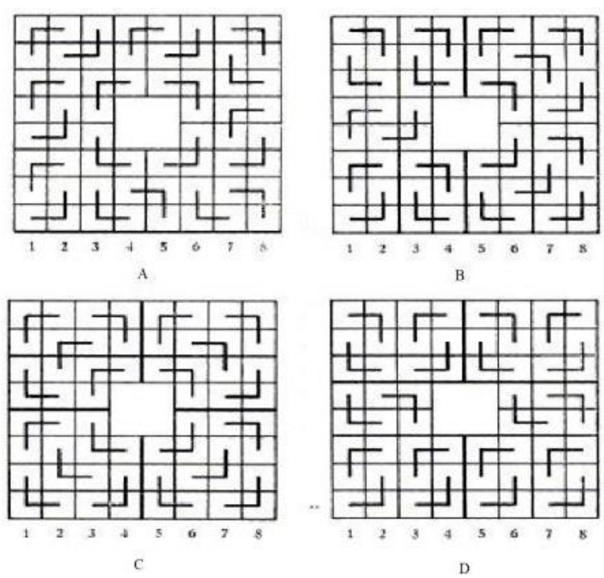


Figure 2 Put tiles

Some studies Applied to dynamic programming Approach for optimal control of dengue virus spread (A.P. Putri, 2018). Visual place recognition in a changing environment (J.H. Oh and B.H. Lee, 2017). Detection of human emotions (W. Mardiniet al., 2018). For the planning of Drone Routes (H. Gjorshevskiet al., 2018). Planning Reliability Growth (L. Sell and W. New, 2018). Electrical vehicle (Z. Pan et al., 2020). The Divide and conquer approach Many works from them Multi-label Learning (W. Zhanget al., 2017). The generation of mid-infrared frequency combs (K.L. Vodopyanov, 2010), Message (I.M. Hayder et al., 2019), Placement of the sensor (C. Jiang, 2019). Rapidly Learning Bayesian (K.L. Vodopyanov, 2010).

Related Work

Researchers have proposed a game model for a strange game, specifically to learn about the interaction between an electrical riot and the attackers on the grid. Lead to correctly track power outages during a non-cooperative, zero-sum attack vector game (Coordinated Cyber-Attacks) CCA (Machine Defector). The study provided for the assignment of infrastructure, defense and cyber security, and the validity of the attack vector was verified by the use of the power of the centered carrier New England 39 bus power with Greedy Method (H. Rp et al., 2019).

This study deals with the teaching arrangement in the divide-and-conquer algorithm, aiming at the deficiency that the new teaching approach uses the ability to decompose sub-problems and raises the complexity of teaching, the coverage series is enhanced for the L-type dominoes. The enhanced algorithm can be compatible with the divide-and-divide strategy Conquering and c and enhance the normative design and accuracy of the process. Algorithm, thus achieving the desired outcome of teaching (Z. Li, F. Huang et al., 2009).

An algorithm was proposed to improve run time to 3.9% based on the results of the implementer's experiments on the proposed algorithm. The algorithm has five stages included in the first stage, the data is stored in the Hadoop structure. In the second stage, we collect data using the MR-DBSCAN-KD method to identify all outliers and clusters.

Outliers are then assigned to the present groups using a futuristic greed method. At the end of the second stage, groups are merged together. Blocks are assigned to the reducers in the third stage. Note that the number of reducers must be reduced for this task by applying an approximate load balancing between the reducers. In the fourth stage, the reducers perform their functions in each group. At the end of the final phase, the output return reducers. The reduction of the number of reducers and the assembly revision helped reducers to do their jobs simultaneously (A. Bakhthemmat & M. Izadi, 2020).

One of the studies showed pure electroencephalography using a dynamic programming approach to detect and classify human emotions in brain waves The inputs of this study are students between the ages of 20-24 years of age who reveal feelings of sadness, happiness and fear by placing sensors on the scalp that transmit signals measured to the sensory device and then analysing positive data results have been obtained (W. Mardiniet al., 2018).

Methods

Dynamic Programming Approach

Divide the problem into a series of overlaps Sub-problems, building solutions to larger and larger Sub-problems and can be store the Sub-problems solutions to solve big Sub-problems, there are two approaches to solve Dynamic Programming the first approach by Bottom-up in this technique divide a problem into Sub-problems then solve children and store it and then.

Solve the parents, the second approach by Top-down approach in this technique divide a problem into Sub-problems then solve parents and store it and then solve the children, in this problem we will use this technique (A. Bakhthemmat & M. Izadi, 2020, H.A. Younis et al., 2020 & A. Ozolins, 2020).

Solve Defective Chessboard Problem by Dynamic Approach

We choose one box of the chessboard and assume that it is at the top right and store the value of the box in the array and then put the shaped like L letter in the center of the chessboard. The angle of the shape depends on the location of the box chosen previously, and then we divided the chessboard into four parts, and we repeat the process by putting the shaped like L letter in the middle of the sub-chessboard, the figure (3) below shows these steps.

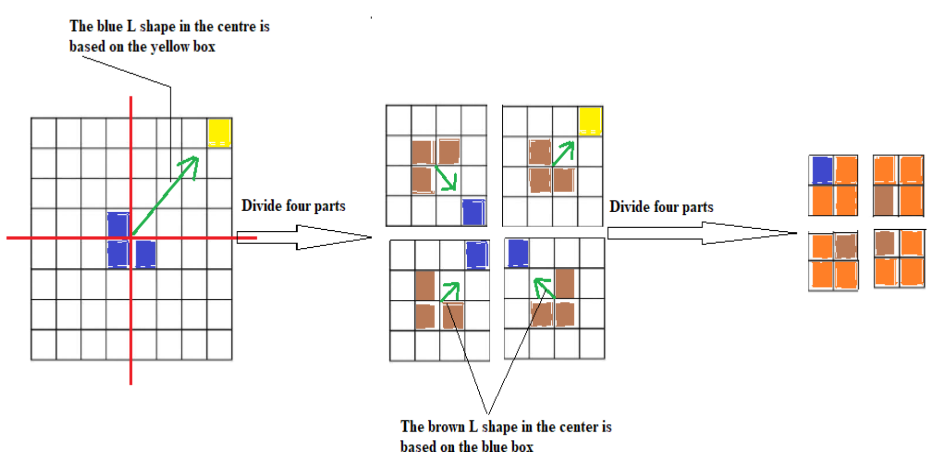


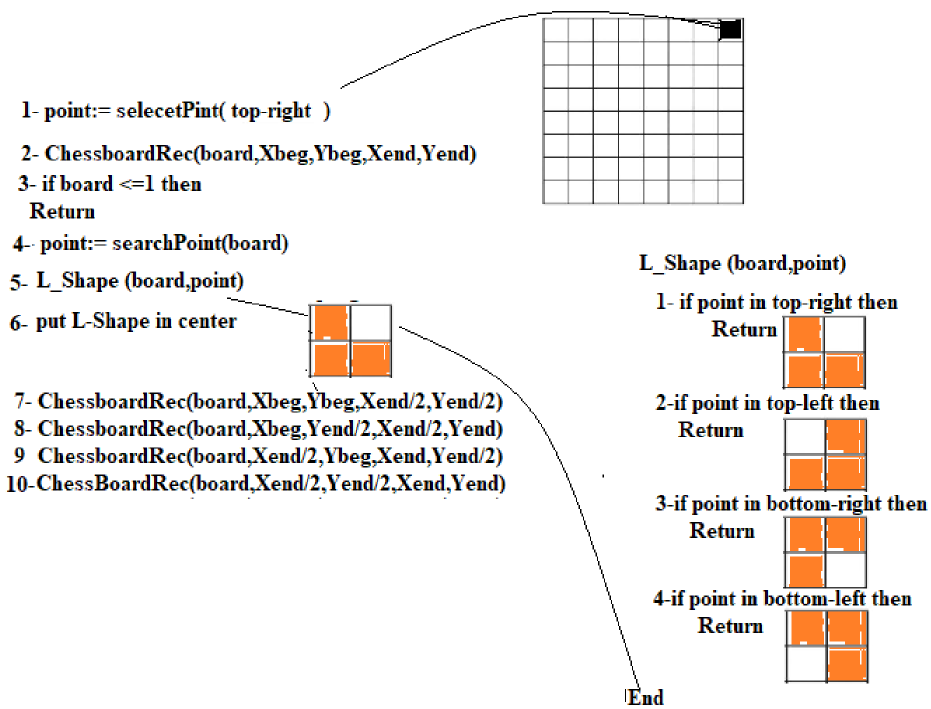
Figure 3 How to solve problem

Proposed Algorithm (1)

Analysis

In this case, classified its dynamic programming because:

1. Each value center save in the array and move to depth by divide chessboard into sub- problem and iteration this process by search the center and then save each point.
2. in this problem overlaps problem because we can't find Children value Unless finding parents value.
3. the new shaped like L letter in the center depends on shaped like L letter on the parents.



$$T(n)=\begin{cases} 0(1) & \text{if } n <= 1 \\ T(n) = 4T(n/2) + c & \text{if } n > 1 \end{cases} \quad (1)$$

Divide & Conquer Approach

It is a way to divide the large problem into two or smaller instances (problems) that are easier to solve, and then merge the solution to solve the big problem. Divide-&-conquer is the strongest algorithm design technique used to solve many important problems such as merge sort and often instances of the original 3 problem and may be solved using the divide- &-conquer strategy recursively. There are also many problems that humans naturally use divide and conquer approaches to solve, such as sorting a stack of playing (A. Bakhthemmat & M. Izadi, 2020), (Z. Li, F. Huang et al., 2009).

Solve Defective Chessboard Problem by Divide-&-Conquer Approach

As mentioned earlier, a divide-&-conquer (DAC) approach is used to solve the problem. DAC entails breaking a big problem into sub-problems, ensuring that each sub-problem is alike of the larger one, albeit smaller. The total number of squares on our board is n^2 or 4^k. Removing the defect, we would have (4^k — 1), which is always a multiple of 3, the figure (4) below show the tree how to divide chessboard.

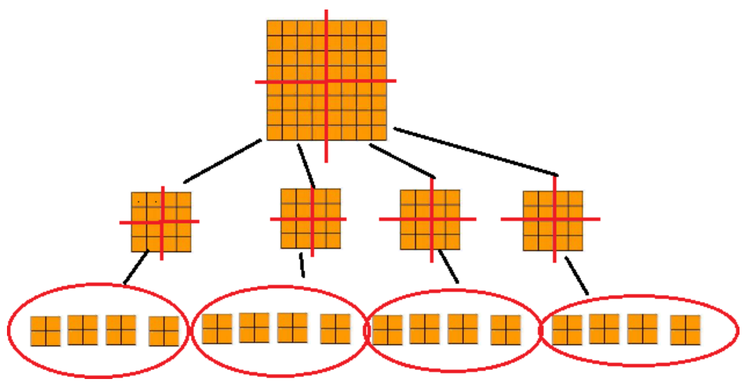


Figure 4 The tree Chessboard Problem by Divide-&-Conquer

Proposed Algorithm (2)

procedure TILE $(N, (B_x, B_y, ((T_x, T_y))$
if $N = 0$ ***then***
return
mid $\leftarrow$ 2^{N-1}
blocked $\leftarrow$ $(0,0), (mid - 1,0), (mid – 1$
$mid - 1, (0, mid - 1)\}$
if $Bx \geq mid$ ***and*** $By \geq mid$ ***then***
blocked Quad $\leftarrow$ ***TOP_RIGHT***
if $B_x < mid$ ***and*** B_y ***mid then***
blocked Quad $\leftarrow$ ***TOP_LEFT***
if $B_x < mid$ ***and*** $B_y < mid$ ***then***
blocked Quad $\leftarrow$ ***BOTTOM_LEFT***
if $B_x > mid$ ***and*** $B_y < mid$ ***then***
blocked Quad $\leftarrow$ ***BOTTOM_RIGHT***
place $(T_x + mid, T_y + mid, blockedQuad)$
tile $(N - 1, blocked\ [0], T_x + mid\ \ T_y + mid)$
tile $(N - 1, blocked\ [1], T_x, T_y + mid)$
tile $(N - 1, blocked\ [2], T.TU)$
tile $(N - 1, blocked\ [3], T_x + mid\ T_y)$

Analysis

After fixing each of the four sub-problem and merging them together to form a complete board, we have 4 flaws in our board: the original defect will fall into one of the quarters, while the other three were the ones we intentionally added to solve the problem with the DAC. All we have to do is add a final Tryonimo to cover these three "flaws" and we're done.

Thus, the recursive equation becomes for time complexity:

$$T(n) = 4T(n/2)+cc \qquad (2)$$

Greedy Method Approach

The greedy algorithm tries to resolve the optimization problem through usually choosing the next optimal step locally. This will generally lead to the optimal solution locally, however no longer necessarily to the optimal solution globally. When our optimization goal is to amplify some quantity, we call the optimum answer locally the maximum and the superior globally max. If we reduce the amount, we call it minimum or minimal respectively. In reality it capability optimization problems we wish to find an optimal solution, among all feasible solutions, to either minimize the cost or maximize (A. Bakhthemmat & M. Izadi, 2020), (J. Sannemo, 2018).

Solve Defective Chessboard Problem by Greedy Method Approach

At the beginning of the algorithm, we will find first for the (main and secondary) diagonal, and then we will divide a big problem into small problems. Each sub-problem has (locally- Optimal). And in the optimization, we will find the (main or secondary) diagonal depending on the location of the square until we reach the end of the tree, the figure (5) below shows algorithm.

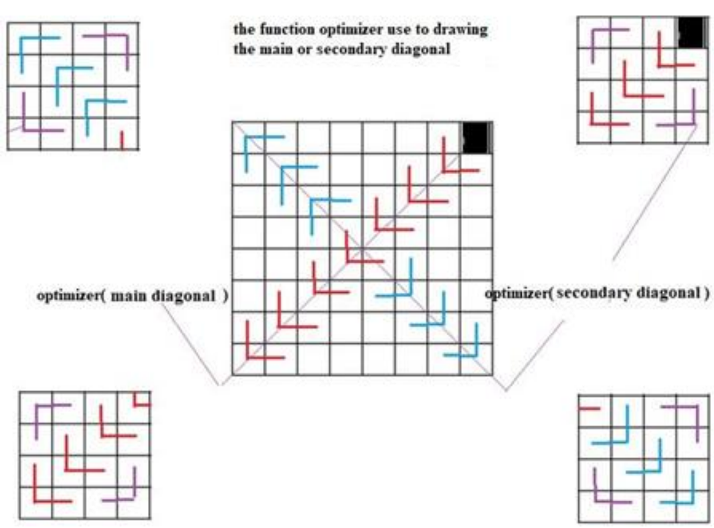


Figure 5 How to find main or secondary diagonal

Proposed Algorithm (3)

1. Draw diagonal (main and secondary)
2. ChessboardRec (board, Xbeg, Ybeg, Xend, Yeud)
3. if board 4 theu Return
4. Draw diagonal (main or secondary)
5. ChessboardRec (board Xbeg, Ybeg, Xend / 2.Yend) / 2)
6. ChessboardRec (board, Xbeg, Yend / 2, Xend / 2.Yend)
7. Chessboard Rec (board, Xend / 2, Ybeg, Xend, Yend / 2)
8. ChessBoardRec (board Xeud / 2.Yend/ 2, Xead , Yend)

Figure 6 Proposed Algorithm 3

Analysis

In this case, classified its greedy method because:

1. Grow the tree gradually, and each sub-problem solve by optimization.
2. In this case, optimization is drawing the main or secondary diagonal depending on the location of the chessboard if it is on the (up-left or bottom-right) side, the optimizer will draw the main diagonal, otherwise will draw secondary diagonal, the figure (6) shows how to work the optimizer. the figure (7) below shows how to analyze the tree.

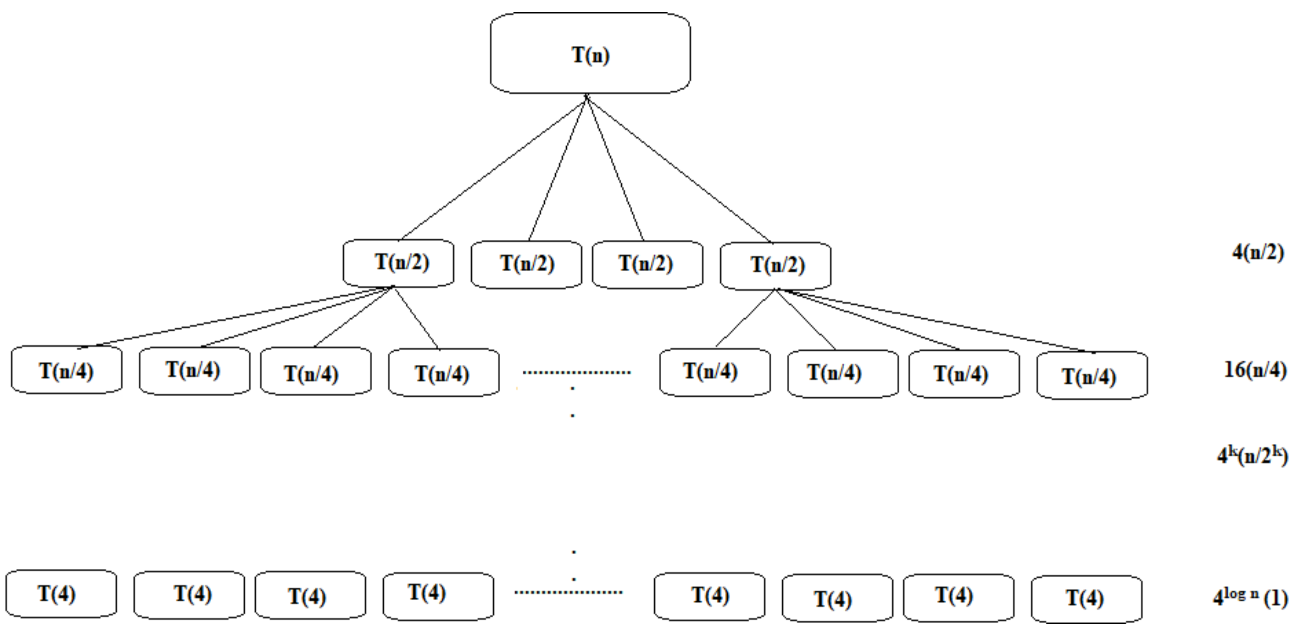


Figure 7 The tile analysis tree

Conclusion

The figure (8) below shows how to fill an L_shaped array with three algorithms and shows the difference between them.

```
[[ 3   3   4   4   8   8   9   0]
 [ 3   2   2   4   8   7   9   9]
 [ 5   2   6   6  10   7   7  11]
 [ 13 13  14   1   1  18  19  19]
 [13  12  14  14  18  18  17  19]
 [15  12  12  16  20  17  17  21]
 [15  15  16  16  20  20  21 21]]
```

1) Dynamic Programming Approach

```
[[1   1   2   2   6   6   7   7]
 [1   5   5   2   6  10  10   7]
 [3   5   4   4   8   8  10   9]
 [3   3   4  21  21   8   9   9]
 [11 11  12  21  16  16  17  17]
 [11 15  12  12  16  20  20  17]
 [13 15  15  14  18  20  19  19]
 [13  13 14  14  18  18  19 -1]]
```

2) Divide &conquer approach

```
[[0   0  13  13  15  15   6  -1]
 [0   1   1  13  15   7   6   6]
 [14  1   2   2   8   7   7  16]
 [14 14   2   9   8   8  16  16]
 [17 17  10   9   9   3  19  19]
 [17 11  10  10   3   3   4  19]
 [12 11  11  1 8 20   4   4   5]
 [12 12  18  18  20  20   5   5]]
```

3) Greedy Method Approach

Figure 8 The filling of an array with L_shaped

In this step, we will implement a program that includes three approaches and compare the difference between time complexity and summary (1). The difference appears when exponential increases to 13. In Divide & conquer it reaches 100 seconds, in the Dynamic Programming reaches 89 seconds, and Greedy Method reaches 33 seconds.

Table 1 The time summary for each approach

Time summary			
	DynamicPro	D & C	GreedyMethod
0.0	0.000041	0.000049	0.000013
1.0	0.000049	0.000061	0.000024
2.0	0.000128	0.000135	0.000034
3.0	0.000335	0.000403	0.000121
4.0	0.001310	0.001298	0.000399
5.0	0.005173	0.005418	0.001745
6.0	0.019932	0.020620	0,006643
7.0	0.081807	0.091426	0.028207
8.0	0.331771	0.367435	0.126042
9. 0	1.393112	1.530192	0.509616
10.0	5.621440	6.356095	2.117079
11.0	22.388280	25.191683	8.000013
12.0	89.661425	100.000013	33.000013

We notice that the greedy algorithm is faster than the rest of the algorithms, then the dynamic algorithm, and finally divided & conquer is slower than the all, and Figure (9) below shows the difference between them.

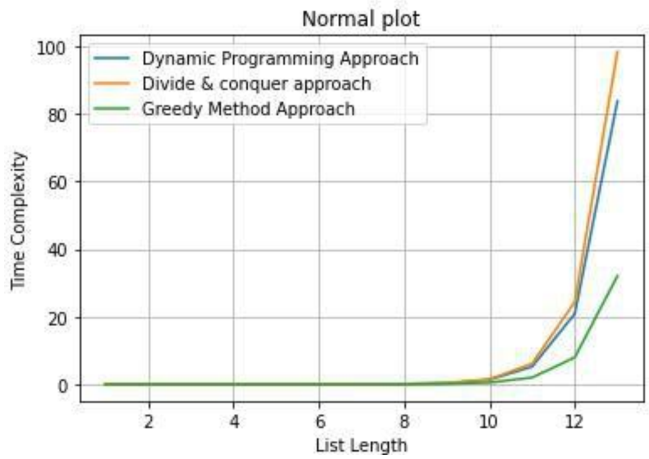


Figure 9 Running Time the three algorithm

References

Levitin, A., & Levitin, M. (2011). Algorithmic puzzles. *Journal of Visual Languages & Computing*, *11*(2559).

Putri, A.P. (2018). Neuro-dynamic programming approach to optimal control of spreading of dengue viruses. *In International Conference on Computer, Control, Informatics and its Applications (IC3INA)*, 169-174.

Oh, J.H., & Lee, B.H. (2017). Dynamic programming approach to visual place recognition in changing environments. *Electronics Letters*, *53*(6), 391-393. https://doi.org/10.1049/el.2017.0037

Mardini, W., Ali, G.A.S., Magdady, E., & Al-momani, S. (2018). Detecting human emotions using electroencephalography (EEG) using dynamic programming approach. *In 6th International Symposium on Digital Forensic and Security (ISDFS)*, 1-5.

Gjorshevski, H., Trivodaliev, K., Kosovic, I.N., Kalajdziski, S., & Stojkoska, B.R. (2018). Dynamic Programming Approach for Drone Routes Planning. *In 26th Telecommunications Forum (TELFOR)*, 1-4.

Sell, L., Guo, J., Li, Z.S., & Keyser, T. (2018). A Dynamic Programming Approach for Planning Reliability Growth. *In Annual Reliability and Maintainability Symposium (RAMS)*, 1-6.

Pan, Z., Yu, T., Li, J., Qu, K., Chen, L., Yang, B., & Guo, W. (2020). Stochastic transactive control for electric vehicle aggregators coordination: A decentralized approximate dynamic programming approach. *IEEE Transactions on Smart Grid*, *11*(5), 4261-4277. http://doi.org/10.1109/TSG.2020.2992863

Zhang, W., Wang, X., Yan, J., & Zha, H. (2017). A divide-and-conquer approach for large-scale multi-label learning. *In IEEE Third International Conference on Multimedia Big Data (BigMM),* 398-401. http://doi.org/10.1109/BigMM.2017.35

Vodopyanov, K.L., Leindecker, N.C., Marandi, A., Byer, R.L., & Pervak, V. (2011). Divide-and-conquer approach to the generation of mid-infrared frequency combs. *In Conference on Lasers and Electro-Optics Europe and 12th European Quantum Electronics Conference (CLEO EUROPE/EQEC),* 1-1.

Ghosh, G., Samanta, D., Paul, M., & Janghel, N.K. (2017). Hiding based message communication techniques depends on divide and conquer approach. *In International Conference on Computing Methodologies and Communication (ICCMC),* 123-128.

Farahat, A.K., Ghodsi, A., & Kamel, M.S. (2011). An efficient greedy method for unsupervised feature selection. *In IEEE 11th International Conference on Data Mining,* 161-170. http://doi.org/10.1109/ICDM.2011.22

Hariharan, R., Mahesh, C., Prasenna, P., & Kumar, R. V. (2016). Enhancing privacy preservation in data mining using cluster based greedy method in hierarchical approach. *Indian Journal of Science and Technology, 9*(3), 1-8.

Guan, P., & Wang, J. (2019). Optimal Adaptive Coordinated Cyber-Attacks on Power Grids using∊-Greedy Method. *In North American Power Symposium (NAPS), 67*(9), 2249-2262. http://doi.org/10.1109/TSP.2019.2903017

Hayder, I.M., Younis, H.A., & Younis, H.A.K. (2019). Digital Image Enhancement Gray Scale Images in Frequency Domain. *In Journal of Physics: Conference Series, 1279*(1). http://doi.org/0.1088/1742-6596/1279/1/012072

Jiang, C., Chen, Z., Su, R., & Soh, Y.C. (2019). Group greedy method for sensor placement. *IEEE Transactions on Signal Processing*, *67*(9), 2249-2262. http://doi.org/10.1109/TSP.2019.2903017

Zhang, W., Feng, W., Zhao, H., & Zhao, Q. (2019). Rapidly Learning Bayesian Networks for Complex System Diagnosis: A Reinforcement Learning Directed Greedy Search Approach. *IEEE Access*, *8*, 2813-2823. http://doi.org/10.1109/ACCESS.2019.2952143.

Guan, P., & Wang, J. (2019). Optimal Adaptive Coordinated Cyber-Attacks on Power Grids using∊-Greedy Method. *In North American Power Symposium (NAPS), 67*(9), 2249-2262. http://doi.org/10.1109/TSP.2019.2903017

Li, Z., Huang, F., Liu, X., & Duan, X. (2010). Chessboard Coverage Teaching Based on Divide-and-Conquer Algorithm. *Modern Applied Science*, *4*(1), 36-43. http://doi.org/10.5539/mas.v4n1p36

Bakhthemmat, A., & Izadi, M. (2020). Decreasing the execution time of reducers by revising clustering based on the futuristic greedy approach. *Journal of Big Data*, *7*(1), 1-21. http://doi.org/10.1186/s40537-019-0279-z

Sannemo, J. (2018). *Principles of Algorithmic Problem Solving.* Draft version.

Younis, H.A., Hayder, I.M., Seger, I.S., & Younis, H.A.K. (2020). Design and implementation of a system that preserves the confidentiality of stream cipher in non-linear flow coding. *Journal of Discrete Mathematical Sciences and Cryptography*, *23*(7), 1409-1419. http://doi.org/10.1080/09720529.2020.1714890

Ozolins, A. (2020). Bounded dynamic programming algorithm for the job shop problem with sequence dependent setup times. *Operational Research, 20*(3), 1701-1728. http://doi.org/10.1007/s12351-018-0381-6

Malhotra, M., & Chhabra, J.K. (2018). Micro level source code summarization of optimal set of object oriented classes. *Webology, 15*(2), 113-132.

Appendices

Appendix (A)

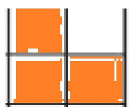
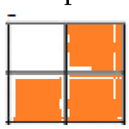
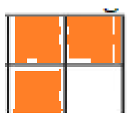
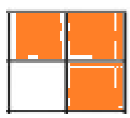
Proposed Algorithm (1)

This method can be used to save data in memory as L-shape to avoid hackers attack, so it is not easy to track data unless having the method in which it was saved previously.

```
K=13
board =(2^k,2^k)
point=selecting point        \\assume top-right inboard is the selected point.
Xbeg ,Ybeg =0
Xend ,Yend =2^k
ChessBoardRec(board,Xbeg,Ybeg,Xend,Yend)
If board <=1 then
    Return
End if
point=SearchPoint(board)
L_Shape=DrawL_shape(board,point)
  Put L_Shape in the center
ChessBoardRec(board,Xbeg,Ybeg,Xend/2,Yend/2)
ChessBoardRec(board,Xbeg,Yend/2,Xend/2,Yend)
ChessBoardRec(board,Xend/2,Ybeg,Xend,Yend/2)
ChessBoardRec(board,Xend/2,Yend/2,Xend,Yend)
```

```
SearchPoint(board)
If board (0,0) <>Empty
    Return point= board (0,0)
Else if board (0, Yend) <> Empty
      Return point= board (0, Yend)
  Else if board (Xend, Yend) <> Empty
```

7. Return point= board (Xend, Yend)
8. Else if board (Xend, 0) <> Empty
9. Return point= board (Xend, 0)
10. End if

1. DrawL_shape(board,point)
2. If point = top-right then
3. **Return** 
4. Else If point = top-left then
5. **Return** 
6. Else If point = bottom-right then
7. **Return** 
8. Else If point = bottom-left then
9. **Return** 
10. End if

Appendix (B)

Proposed Algorithm (3)

1. K=13
2. board =(2k,2k)
3. DrawDiagonal (main and secondary)
4. ChessBoardRec (board,Xbeg,Ybeg,Xend,Yend)
5. If board <= 4 then
6. Return
7. End if
8. DrawDiagonal (main or secondary)
9. ChessBoardRec(board,Xbeg,Ybeg,Xend/2,Yend/2)
10. ChessBoardRec(board,Xbeg,Yend/2,Xend/2,Yend)
11. ChessBoardRec(board,Xend/2,Ybeg,Xend,Yend/2)
12. ChessBoardRec (board,Xend/2,Yend/2,Xend,Yend)