MedCrave
Step into the World of Research

International Robotics & Automation Journal

Research Article

Open Access

CrossMark

Lightbot code hour: optimal solutions, theory and designs

Abstract

Lightbot is an interesting game app that has strong educational purpose to guide kids at young age to learn programming skills. It has a free version named Lightbot Code Hour consisting of 20 levels. The optimal solutions were studied recently. In this paper, we improved the previously published result of 174 blocks to 173 blocks and proved that this is the optimal solution that can no longer be further improved. A theoretical bound for number of periods of a finite map is given. A sharp bound is given for a simplified model. An efficient stopping criteria is presented. Some interesting designs are also presented to demonstrate complicated solutions with large number of periods.

Keywords: lightbot, optimal solution, periodicity, symmetry

Volume 11 Issue 1 - 2025

Songming Hou,[1] Jianning Su,[2] Yan Huang[3]
[1]Program of Mathematics and Statistics, and Center of Applied Physics, Louisiana Tech University, USA
[2]Department of Mathematics, Computer Science, and Engineering, Georgia State University Perimeter College, USA
[3]Beijing AndEngine Software, China

Correspondence: Songming Hou, Program of Mathematics and Statistics, and Center of Applied Physics
Louisiana Tech University, Ruston, Louisiana, 71272, USA

Received: April 12, 2025 | **Published:** April 25, 2025

Introduction

Around 30 or 40 years ago, most students and kids before entering college did not have access to computers regularly.[1,2] The situation has completely changed. Many families have at least a computer at home.[2] Further, electronic devices such as iPhone and iPad can almost serve as a computer. Most students and kids at young age do have access to such devices. However, just because they have such access does not mean they all have the interest learning programming skills. On the other hand, most of them are interested in playing games. Extensive studies have been done on serious-game designs.[3–5] In particular, Lightbot is a good tool to motivate students and kids at young age to learn programming skills through playing game to give them a head start entering college. It introduces programming concepts like loops and subroutines. It has a free version named Lightbot Code Hour consisting of 20 levels. The goal is to light up all the lights using a robot by giving it fixed instructions. It has been studied in.[6,7] However, the optimal solutions are not discussed until the recent work[8] by the first and third author. In that paper, the optimal solutions found add up to 174 blocks. Unfortunately, that is not really optimal. The issue was the lack of a rigorous way to terminate the search.

When looking for the optimal solutions for Lightbot levels, the idea of periodicity comes into play which is shared by another game named the Rubik's Snake invented over 40 years ago.[9] The applications of the Rubik's Snake include the study of protein folding[10] and for the construction of reconfigurable modular robots.[11–13] Some mathematical problems including periodicity concerning a Rubik's Snake have been studied.[14] In,[15] theorems about MÖbius, palindromic and periodic Rubik's snakes were proved. Further, the study of shortest path of Rubik's Snake knots[16–20] also utilized periodicity and share the same scheme of optimization with this work.

The rest of the paper is organized as follows: in Section 2, we give a simple tutorial how to play Lightbot Code Hour and a complete list of number of blocks needed, adding up to 173 blocks. In Section 3, we explain level 2-6 in details, which we saved a block compared with.[8] In Section 4, we explain a theoretical bound and an efficient stopping criteria, with which we rigorously proved the optimal solution is 173 overall. In Section 5, we present a sharp theoretical bound for a simple model. In Section 6, we present complicated designs with large number of periods needed to have an optimal solution. We conclude in Section 7.

A simple tutorial for lightbot code hour

Lightbot Code Hour can be downloaded for free from app stores on one's iphone or ipad. From level 1-1 to 1-8, one only need to add icons representing forward, left turn, right turn, jump, and light to the main program. The blue spots are lights that are off initially. A robot is initially at certain position with certain orientation. It cannot go out of bound. It cannot jump to more than one level above or jump forward to the same level. It cannot go forward if the forward direction has uneven surface. If the robot is on a light spot and the light command is used, then the light will be switched from off to on or from on to off (yellow means on and blue means off). When all lights are on, the puzzle is solved and the robot stops even if there are more commands not processed. In Stage 1, since we only have a main program and the only way is to step-by-step follow the commands, the solution is obvious and can be easily optimized to have the smallest number of blocks. In Stage 2, subroutine is introduced. In Stage 3, the concept of loop is the focus, though the only difference compared with Stage 2 is it emphasized loop by allowing only one block in the main program.

We present a complete list of the number of blocks needed for optimal solutions below. What is new compared with[8] is, first, level 2-6 is improved from 11 to 10 blocks. Second, this time we used a stopping criteria to efficiently and rigorously proved the optimal solution.

For Stage 1 levels, the number of blocks are 3+9+6+8+11+12+12+12=73.

For Stage 2 levels, the number of blocks are 7+7+10+14+10+10=58.

For Stage 3 levels, the number of blocks are 4+4+8+7+10+9=42.

Overall, 73+58+42=173 is the optimal solution. Figure 1 shows a screen capture proving that we have reached this solution. To our knowledge, this was not known in the literature (Figure 1).

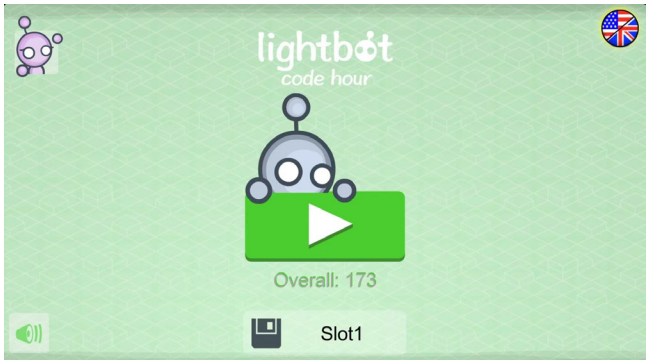

Figure 1 The overall optimal solution.

Level 2-6

We cannot let the loop run forever to verify solutions. In,[8] by experience, 20 periods were set as the maximum. This works for all levels except 2-6, which actually needed at least 22 periods to reach an optimal solution. Since we do not know in advance the exact number of periods needed, a stopping criteria is required and we will discuss that in the next section. Here we present the new result of 2-6 with only 10 blocks. During the procedure, we take 4 periods to reach the center with all lights off (some on and then off). Then we have the best symmetry. Each 5 periods let the robot rotate by 90 degrees and go back to the center. A quarter of the total number of lights are on. Repeat 4 times. There is a minor detail that the last time we only take 3 periods and the lights are all on. Overall we have 4+5+5+5+3=22 periods. Figure 2 shows the commands for the optimal solution and Figure 3 gives the detailed explanation how the robot moves in 22 periods (all lights are on before the 22th period finishes and the robot stops at the location labeled as 22 with the orientation shown) (Figures 2&3).

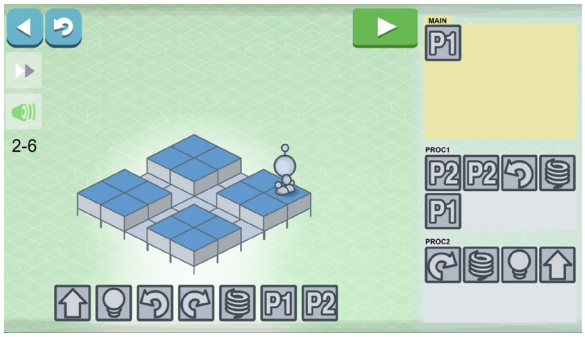

Figure 2 Level 2-6.

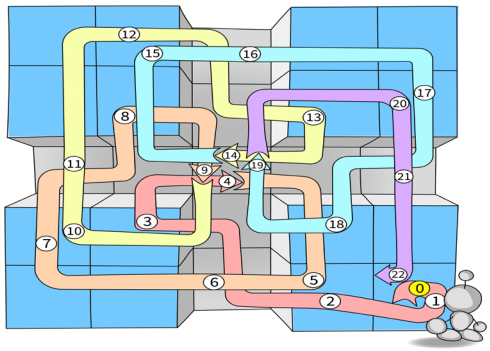

Figure 3 Level 2-6 explained.

Theoretical bound and an efficient stopping criteria

Assume each unit square is of unit length. Suppose the map viewed from the top has area A. Within $4A$ periods, we have $4A + 1$ states (in terms of location and orientation of the robot) at the beginning or end of a period. On the other hand, there are only $4A$ different states if only counting the current location and orientation. Therefore there must exist a pair of states that are identical in terms of location and orientation. From the first of these two to the second, keep track the change of the light status. If we repeat this many periods another time, all the changes of light status are doubled, and therefore unchanged. This implies that within $8A$ periods, there must exist two state at the beginning or end of a period such that they are identical in terms of location, orientation and all the status of lights. This means $8A$ is the theoretical bound for a loop, as further moves can only repeat history.

In practice, we use the following efficient criteria to terminate a loop: keep track of history. If all lights are on, immediately terminate and do not even need to finish the current period. At the end of each period, if the location and orientation of the robot as well as all the status of lights is a repeat of certain history, then terminate the loop. There is no need to count to a theoretical bound of $8A$, as in practice normally it takes much fewer than $8A$ periods. This stopping criteria on one hand ensures we do not miss any valid solution, on the other hand has practical cost.

For each level in Lightbot Code Hour, we test the minimal number of blocks with an infinite loop and the minimal number of blocks without an infinite loop. Start with the search for a solution with one block then add more blocks. If one of the two situations (with or without an infinite loop) has a solution for a block number, we stop the search. By doing so, we are able to find all optimal solutions rigorously. 173 is indeed the optimal number of blocks that cannot be further improved. Among all levels, 2-4 is the only level that we used the solution found by us in the previous paper and only checked that there is no solution with less than 14 blocks, proving we found the optimal solution. For all other levels, the computer code found the optimal solutions. It is just a matter of computational cost.

A sharp theoretical bound for a simple model

Although the bound of $8A$ is correct, it is not a sharp bound. In practice the number of periods are normally much smaller. The problem is too complicated to look for a sharp bound. Therefore we present a simple model with an $m - by - n$ flat map and prove a sharp bound under certain assumptions.

We label each point (i, j) with i between 0 and $m - 1$ and j between 0 and $n - 1$ for the possible locations for the robot. Consider Main with only one block "P1" and PROC1 with only one "P1" and is at the end. "P2" is not used. Furthermore, assume the operation sequence in PROC1 is a sequence with sum of rotation degrees to be odd integer multiple of 90 degrees. We make this assumption because this is the non-trivial case. If the sum of rotation degrees is equivalent to 0 degree, then the bound for number of periods depends on the size. We could simply sweep row by row or column by column, ending with the same orientation each time. If the sum of rotation degrees is equivalent to 180 degree, then the bound for number of periods is even smaller than the case we discuss because the robot only has two variations of orientation at end of periods.

We track the positions and orientations when repeating PROC1. Let (p_i, o_i) be the sequence of positions, p_i, and orientations, o_i, after repeating PROC1 i times for $i = 0, 1, 2, 3, \ldots$ Let $p_i = (x_i, y_i)$ for all i.

Citation: Hou S, Su J, Huang Y. Lightbot code hour: optimal solutions, theory and designs. *Int Rob Auto J.* 2025;11(1):19–23.
DOI: 10.15406/iratj.2025.11.00295

Lemma. (1). The sequence $(xi)_{i=4}^{\infty}$ is periodic with period of 4. (2). If $m = 1$, then $(xi)_{i=0}^{\infty}$ is periodic with period of 4. (3). If $m = 2$, then $(xi)_{i=2}^{\infty}$ is periodic with period of 4.

Proof. To prove (1), we just need to prove $x8 = x4$.

We assume that the length of PROC1 is k. From the initial position (x_0, y_0), we now focus on applying PROC1 four times. Note that there are $4k$ moves. For each $i = 1, 2, ..., 4k$, define c_i as follows: $c_i = 1$ if the operation is forward to the right; $c_i = -1$ if the operation is forward to the left; $c_i = 0$ otherwise. This represents, in general, the change amount of the x-coordinate for each move, except the case when the robot hits a boundary. Note that $c_{i+2k} = -c_i$ for all $1 \le i \le k$. Hence, the total sum is 0. It means $x_4 = x_0$ if the robot does not hit a boundary. Assume that the robot hit (that is, attempting to walk out of bound but getting stuck) the left boundary a times at $x = 0$ and the right boundary b times at $x = m - 1$. Then $x_4 = x_0 + a - b$. We consider the following three cases.

Case 1. Assume there exists i, j such that $| c_i + c_{i+1} + ... + c_j | \ge m - 1$. Without loss of generality, we may assume $c_i + c_{i+1} + ... + c_j \ge m - 1$. Then, no matter what starting position is given, the robot reaches the right boundary at $x = m - 1$ after j moves. Due to this, the movement after that is always the same and so the ending position is always the same. Hence, by choosing the starting position to be x_0 and x_4, respectively, we get the ending position $x_4 = x_8$.

Case 2. Assume that both $a = 0$ and $b = 0$. In this case, $x_4 = x_0$. By performing PROC1 four more times, we get $x_8 = x_4$.

Case 3. Assume Case 1 and Case 2 are not satisfied. Then the robot will hit one boundary but not the other. Without loss of generality, we may assume that the robot hit the left boundary a times but does not hit the right boundary. When the robot hits the left boundary, the starting position can be changed from x_0 to $x_0 + 1$ without affecting the movement after the original hit and therefore without affecting the ending position. Note that the robot will not hit the right boundary after the change otherwise it would become Case 1 By repeating the argument, we can change the starting position from x_0 to $x_0 + a$ without affecting the ending position. So, $x_4 = x_0 + a$. Now by performing PROC1 four more times, we get $x_8 = x_4$.

This completes the proof for part (1). Part (2) is trivial because $x_4 = x_0 = 0$.

For part (3), we just need to prove $x_6 = x_2$. We can revise the proof in part (1) by focusing on performing PROC1 two times instead of four times. If PROC1 has no "forward" operation, than it is trivial that $x_6 = x_2 = x_0$. Otherwise, Case 1 always happens and so no matter where it starts (x_0 or x_4), the ending position is the same, i.e., $x_2 = x_6$. This completes the proof for the lemma.

Based on the lemma above, we have the following theorem:

Theorem. Consider an m–by–n flat map. Assume Main has only one block "P1" and PROC1 with only one "P1" and is at the end. "P2" is not used. Furthermore, assume the operation sequence in PROC1 is a sequence with sum of rotation degrees to be odd integer multiple of 90 degrees. Then the sequence $(p_i, O_i)_{i=4}^{\infty}$ is periodic with period of 4. The Lightbot game will either stop within 11 repetitions or will never stop.

Proof. By the lemma, $(xi)_{i=4}^{\infty}$ is periodic with a period of 4. Similarly $(yi)_{i=4}^{\infty}$ is periodic with a period of 4 and so $(p_i, O_i)_{i=4}^{\infty}$ is periodic with a period of 4. For the Lightbot game, let S_i be the status (including robot position, orientation, and lights on/off status) after applying PROC1 i times. Then $S_4 = S_{12}$ as the lights being switched from S_4 to S_8 are switched again from S_8 to S_{12} and so changed back to the status of S_4. Repeating the argument, we have $S_i = S_{i+8}$ for $i \ge 4$. Therefore, if the Lightbot game does not stop within 11 repetitions, then it will repeat status S_4 to S_{11} forever and never stop. This completes the proof.

We present an example with 11 periods under the same assumption as the above theorem. The main program calls P1 with "light, forward, right, right, forward, forward, right, P1" in P1. Figure 4 shows the first 4 periods and the last 7 periods. Yellow lights mean that they are on after the first 4 periods. The 11th period only executed the light command and then stopped, based on the rule of the game.

This example also better explains the lemma. A key idea in the lemma is counting the number of hits (that is, the forward command not working because of the boundary) at one boundary. On the left of Figure 4, we could see that the robot bumps into the left wall at the beginning of the third period and the lower wall at the beginning of the fourth period. This is Case 3 in part (1) of the lemma (the robot hit the left boundary one time but not the right boundary) with a = 1. By applying the lemma once in x direction and once in y direction, the net effect is that the robot is shifted one unit to the right and one unit up after four periods, which is consistent with what we observed in the figure after four periods (Figure 4).

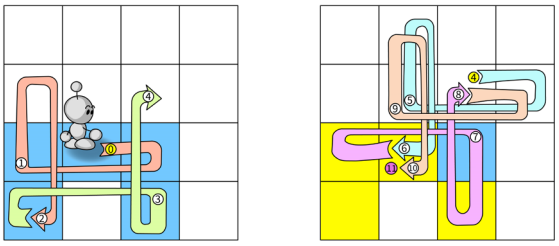

Figure 4 The first 4 and last 7 periods with "light, forward, right, right, forward, forward, right, P1" in P1 called by main.

Complicated designs with large number of periods needed for the optimal solution

Now we turn our role from gamer to designer. Level 2-6 can have 22 or 23 periods for different designs of optimal solutions. Other levels have less periods for optimal solutions. Can we construct some puzzles with a larger number of periods needed for the optimal solution?

The idea is as follows. First, choose some line segment with length not all the same but close. Second, design a path with such line segments such that each turn is 90 degree to the left. Third, let the number of segments be odd. In the end, we are mapped to somewhere with orientation changed by 90 degree times an odd number. Repeat 4 times, the shifts automatically cancel each other (assuming we are not hitting the boundary) and the orientation is back to original. For simplicity, put lights together with each left turn. Note that if a same spot has even number of left turns on it, we do not put a light, otherwise it will be hit even number of times. Finally, design some blocks one level higher and design a sequence in one period such that it works for all line segments.

We present two examples of designs. In Figure 5, we designed an $8 - by - 8$ map that has an optimal solution with 9 blocks and with period 28, which is 4 times an odd number. The solution is to call P1 from Main then have "forward, forward, jump, forward, forward, left,

Citation: Hou S, Su J, Huang Y. Lightbot code hour: optimal solutions, theory and designs. *Int Rob Auto J.* 2025;11(1):19–23.
DOI: 10.15406/iratj.2025.11.00295

light, P1" in P1. In Figure 6, we designed a $9-by-9$ map that has an optimal solution with 9 blocks and with period 36, which is 4 times an odd number. The solution is to call P1 from Main then have "forward, jump, jump, forward, jump, left, light, P1" in P1. For both designs, the beauty of symmetry and periodicity is demonstrated. The length of the paths for each period are not necessarily the same due to the difference in local maps. We rigorously verified that these are indeed the optimal solutions by using exhaustive search.

Note that the optimal solutions are not unique. There are other ways to solve and tie for 9 blocks for each of the two designs. The number of periods before all lights are on could be changed, too. These make the designs more interesting, as players can explore some variations. Even if players are unable to find optimal solutions, by deigning a path to cover a quarter of the lights then repeat, players can finish with more blocks provided that there is no strict limitation for the number of blocks in each subroutine (Figures 5&6).

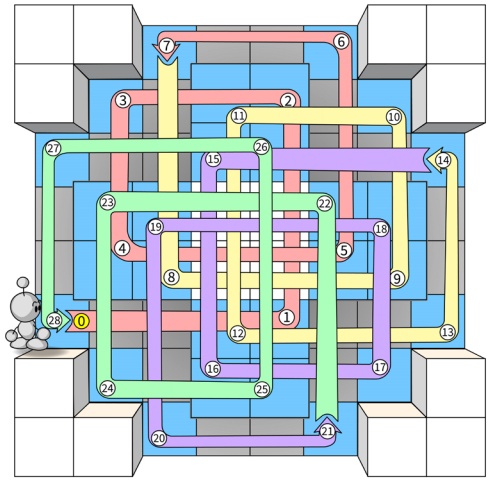

Figure 5 The optimal solution with 28 periods of "forward, forward, jump, forward, forward, left, light, P1" in P1 called by main.

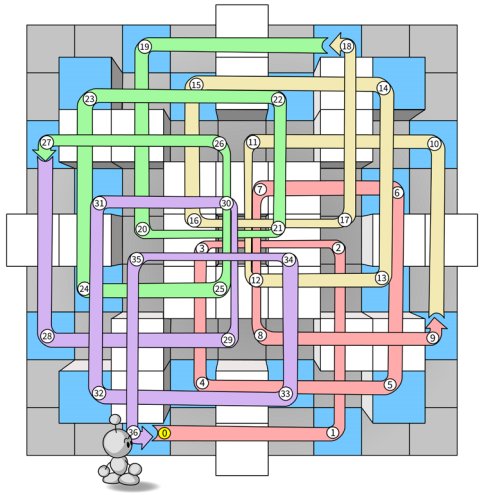

Figure 6 The optimal solution with 36 periods of "forward, jump, jump, forward, jump, left, light, P1" in P1 called by main.

Conclusion

Although Lightbot Code Hour is a game for kids to learn programming skills, finding optimal solutions and designing interesting levels are non-trivial tasks. We proved some theoretical bounds and proposed a practical stopping criteria to ensure the solutions we found are indeed optimal, that 173 is the overall optimal solution, instead of the previously published 174. We also provided some interesting designs for new levels.

Acknowledgments

We thank Yin Sun, Mingjia Yang and Jie Zhu for some interesting discussions. S. Hou's research is partially supported by the Walter Koss Endowed Professorship. The title of the professorship is made available through the State of Louisiana Board of Regents Support Funds.

Conflicts of interest

Authors declare that there is no conflict of interest.

References

1. Households with a computer 40 years ago. 2021.
2. Households with a computer now. 2024.
3. Akkaya A, Akpinar Y. Experiential serious–game design for development of knowledge of object–oriented programming and computational thinking skills. *Computer Science Education.* 2022;32(4):476–501.
4. Arif YM, Ayunda N, Diah NM, et al. A systematic review of serious games for health education: Technology, challenges, and fu–ture directions. *Transformative Approaches to Patient Literacy and Healthcare Innovation*. 2024. p. 20–45.
5. de Carvalho CV, Coelho A. Game–based learning, gamification in education and serious games. *Computers.* 2022;11(3).
6. Gouws LA, Bradshaw K, Wentworth P. Computational thinking in educational activities: an evaluation of the educational game light–bot. *In Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education. ITiCSE '13*, 2013. p. 10–15.
7. Gil A, Figueira T, Netto JFM. Extracting learning analytics from lightbot gameplay sessions. *XXIII Simposio´ Brasileiro de Jogos e Entretenimento Digital (SBGames 2024) – Manaus/AM*. 2024.
8. Hou S, Huang Y. Optimal solutions for lightbot code hour. *International Robotics and Automation Journal.* 2025;11(1):8–11.
9. Fenyvesi C. Rubik's snake of 'infinite possibilities. *The Washington Post*. 1981.
10. Iguchi K. A toy model for understanding the conceptual framework of protein folding: Rubik's magic snake model. *Mod Phys Lett B.* 1998;12(13):499–506.
11. Ding X, Lu S, Yang Y. Configuration transformation theory from a chain–type reconfigurable modular mechanism–rubik's snake. *The 13th World Congress in Mechanism and Machine Science*. 2011.
12. Zhang X, Liu J. *Prototype design of a rubik snake robot*. *Mechanisms and Machine Science*. 2016;36.
13. Liu J, Zhang X, Zhang K, et al. Configuration analysis of a reconfigurable rubik's snake robot. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science.* 2019;233(9):3137–3154.
14. Hou S, Chen Y, Li Z. Some mathematical problems related to the rubik's snake. *J Mech Rob.* 2021;13(1):014502.

Citation: Hou S, Su J, Huang Y. Lightbot code hour: optimal solutions, theory and designs. *Int Rob Auto J.* 2025;11(1):19–23.
DOI: 10.15406/iratj.2025.11.00295

15. Hou S, Su J, Chen Y. Palindromic, periodic and mÖbius rubik's snakes. *International Robotics and Automation Journal.* 2021;**7**(3):84–88.

16. Hou S, Su J. Shortest paths of trefoil knot designs using rubik's snakes. *International Robotics and Automation Journal.* 2022;8(1):18–20.

17. Hou S, Su J. Shortest paths of rubik's snake prime knots up to 5 crossings. *International Robotics and Automation Journal.* 2022;8(2):47–50.

18. Hou S, Su J, Mufutau R. Shortest paths of rubik's snake prime knots with up to 6 crossings and application to roller coaster design. *International Robotics and Automation Journal.* 2023;9(1):30–33.

19. Hou S, Su J, Mufutau R. Shortest paths of rubik's snake composite knots up to 8 crossings. *International Robotics and Automation Journal.* 2023;9(3):110–113.

20. Hou S, Su J, Mufutau R. Shortest paths of rubik's snake composite knots with 9 crossings. *In–ternational Robotics and Automation Journal.* 2024;10(1):25–30.

Citation: Hou S, Su J, Huang Y. Lightbot code hour: optimal solutions, theory and designs. *Int Rob Auto J.* 2025;11(1):19–23.
DOI: 10.15406/iratj.2025.11.00295