MedCrave
Step into the World of Research
International Robotics & Automation Journal

Research Article
Open Access
CrossMark

Optimal solutions for the first three stages of paid version of Lightbot

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 we studied in the past. It also has a paid version consisting of 50 levels with some more complicated rules. Although it is not too hard to solve the puzzles, finding the optimal solutions with the smallest number of blocks is non-trivial for some levels. In this paper, we provide a complete list of the block numbers for optimal solutions for the first three stages in the paid version, as well as the detailed analysis of all the non-trivial levels in these three stages.

Keywords: Lightbot, optimal solution, periodicity

Volume 11 Issue 3 - 2025

Songming Hou,[1] Jianning Su[2]
[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

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

Received: October 16, 2025 | **Published:** October 29, 2025

Introduction

Around 30 or 40 years ago, most kids did not have the luxury to access computers daily.[1,2] The situation has changed significantly. Many families have at least a computer at home now.[2] Further, there are several other electronic devices that can serve almost as a computer. Kids at young age do have access to such devices. However, they might not be interested in 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 an advantage entering college. It introduces programming concepts like loops and subroutines. It has a free version named Lightbot Code Hour consisting of 20 levels and a paid version with 50 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, optimal solutions are not discussed until the recent work.[8,9] Optimal solutions found add up to 173 blocks after a rigorous check by the computer for the free version. The paid version was not systemically studied in the literature.

When looking for 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.[10] The applications of the Rubik's Snake include the study of protein folding[11] and for the construction of re-configurable modular robots.[12] Some mathematical problems including periodicity concerning a Rubik's Snake have been studied.[13,14] A comic book was also published to make some basic research accessible for kids.[15]

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. We borrow some ideas from the study of Rubik's snake to carefully study the optimal solutions for Lightbot paid version and provide a complete list of the block numbers for optimal solutions of the first three stages, as well as the detailed analysis of all the non-trivial levels in these stages.

The difference between the paid version and the free version is that several new rules are introduced. In Stage 3 of the paid version, the elevator and the transport are introduced. More complicated rules were introduced in later stages. These make finding the optimal solution a lot more challenging.

The rest of the paper is organized as follows: in Section 2, we give a simple tutorial about how to play Lightbot. In Section 3, we give a complete list of number of blocks needed for the first three stages of the paid version. In Section 4, we explain level 2-6 in details. In Section 5, we ex-plain level 2-8 and 2-9, which are related. In Section 6, we explain levels 3-2, 3-4 and 3-9 together, as they all use the idea of cells with shifts. In Section 7, we explain level 3-3, which uses the idea of cells with rotations. In Section 8, we explain level 3-8, which involves both elevators and transports. We conclude in Section 9.

A simple tutorial to how to play Lightbot

Lightbot can be downloaded from Apple app stores on one's iphone or ipad with a cheap price by searching the keyword "lightbot". After that, simply open the app and start with Stage 1. From level 1-1 to 1-9, one only needs 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 at certain position with certain orientation. It cannot go out of bound. It cannot jump to more than 1 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 blue spot and the light command is used, then the light will be switched from off to on or from on to off. Yellow color represents that a light is on. When all lights are on, the puzzle is solved and the robot stops even if there are more commands not processed. The game designers encourage players to optimize solutions in the sense of having the smallest number of blocks. (Note that the distance traveled is not a concern according to this rule.) Here a block could be "forward", "light", "left", "right", "jump", "P_1" or "P_2", where the last two mean calling the corresponding subroutine. If a player finish all 50 levels, the total number of blocks used will be shown. This attracts players to keep playing the game and improve the solutions with smaller number of blocks. For 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.

From level 2-1 to 2-9, procedures (subroutines) are introduced. They are labeled as P_1 and P_2. The main program can call them and they can call each other or even oneself.

The name for level 3-1 to 3-9 is "overload". In this stage, elevators and transports are introduced. An elevator is activated by a light command when the robot is on it, and it will either go up two layers or

go down four layers if it reaches the top. At a transport location with green color, the robot is moved to a linked location without changing orientation by a light command.

A complete list of the number of blocks needed for optimal solutions for the first three stages

In this section, we provide a complete list of the number of blocks needed for optimal solutions. This might be helpful for readers who do not know if there is room to improve their own solutions. All these optimal solutions are verified by our computer code that they do solve the problems and they cannot be further improved.

For Stage 1 levels, the number of blocks are 3+7+8+6+7+11+12+12+12=78.

For Stage 2 levels, the number of blocks are 8+12+16+10+8+9+9+11+13=96.

For Stage 3 levels, the number of blocks are 9+11+9+10+4+8+9+8+11=79.

Stage 1 is trivial. To our knowledge, the overall optimal results for the second and third stages are not reported in the literature. Players might waste around 10 to 20 blocks total in Stage 2 and 30 to 60 blocks total in Stage 3. As a general comment, for Stages 2 and 3, normally we have more than one solution that tie for an optimal solution with the smallest number of blocks. We only present one for each level. Also, as mentioned in the introduction, we borrow the idea of periodicity from some early work. Here what we mean by a period is a series of actions (left or right or forward or jump or light) to be repeated several times. Note that there are situations that the robot is not showing the same moves. For example, if we repeat the forward command 3 times but the robot hits the boundary after the first two commands, it will no longer move forward. Still, this counts as repeating the same series of commands 3 times.

Level 2-6

This is the first non-trivial level of stage 2 if we do not count 2-4 which has been solved in[8] with 10 blocks as an optimal solution. The idea is that we can arrive at the same situation with a rotation of 90 degrees. Some lights are on, off then on. Figure 1 shows an optimal solution to the problem. Effectively it sets up a repeated sequence of "forward, right, forward, light, left, forward, light, left". Figures 2 & 3 show the middle steps after 3 and 3 + 5 = 8 periods of the above sequence. It is clear that after 3 + 5 * 3 = 18 periods (the 18th period is partially executed), all lights are on.

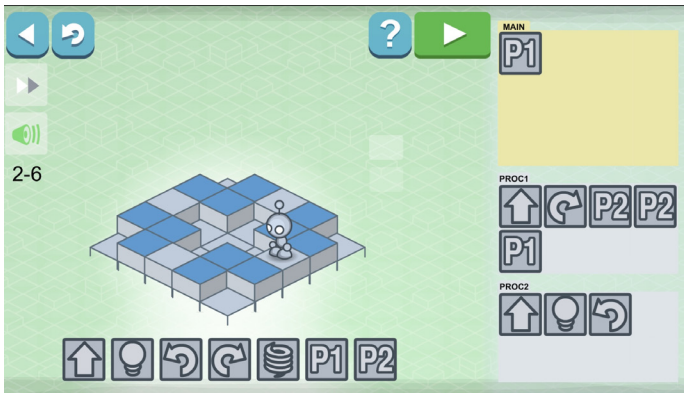

Figure 1 Level 2-6.

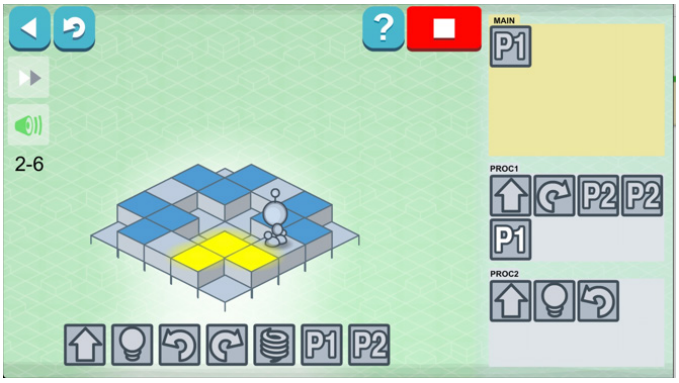

Figure 2 One middle step of 2-6.

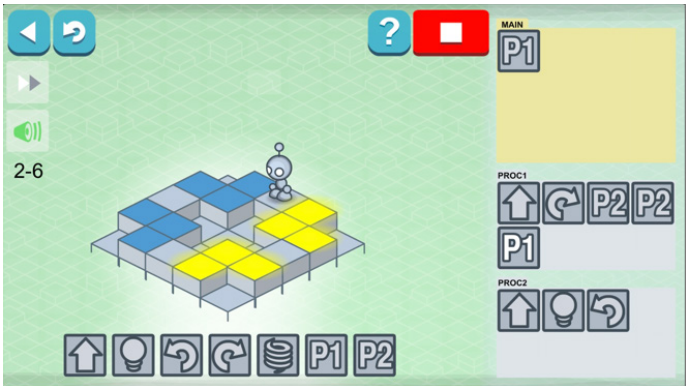

Figure 3 Another middle step of 2-6.

Level 2-8 and 2-9

In level 2-8, if viewed from above, we see a diagram that looks like the number 8, that is, a rectangle with a path in the middle. We can regard this as six vertices and seven edges. We could apply "right, light" on each vertex (we call this "1") and "jump, light, jump, light, jump on each edge (we call this "2"). Now we have a pattern of "1212121221212". To make it periodic, we modify it to be "122122122122122122". Note that adding extra "2" happen not to affect the outcome, as two light commands cancel each other and jump commands will not move the robot at the boundary. The last "2" is not executed as all lights are already on.

Figure 4 shows the result with 11 blocks.

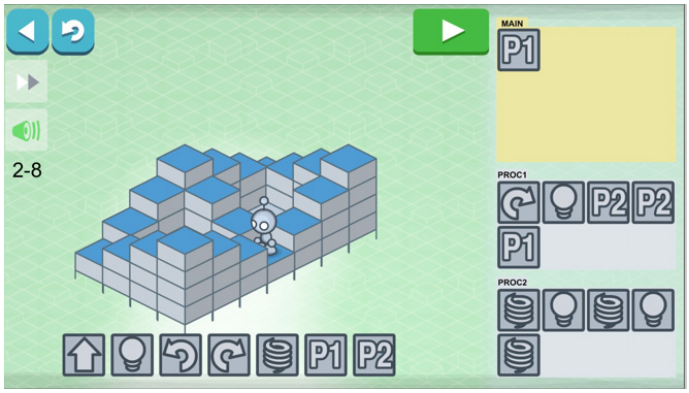

Figure 4 Level 2-8.

Assume we perform "jump, forward, jump" first in main for level 2-9. Then we reach exactly the same situation as level 2-8. In some

Citation: Hou S, Su J. Optimal solutions for the first three stages of paid version of Lightbot. *Int Rob Auto J.* 2025;11(3):98–102.
DOI: 10.15406/iratj.2025.11.00307

sense we can say these two are isomorphic. We then simply copy the idea in 2-8 to solve 2-9. The only thing different is what we call "2" is now "jump, light, forward, light, jump".

Figure 5 shows the result with 14 blocks. However, this is not an optimal solution. The issue is too many blocks are wasted at the beginning. Is there a way to reuse what is in the subroutine for the starting part? We reexamine what we have.

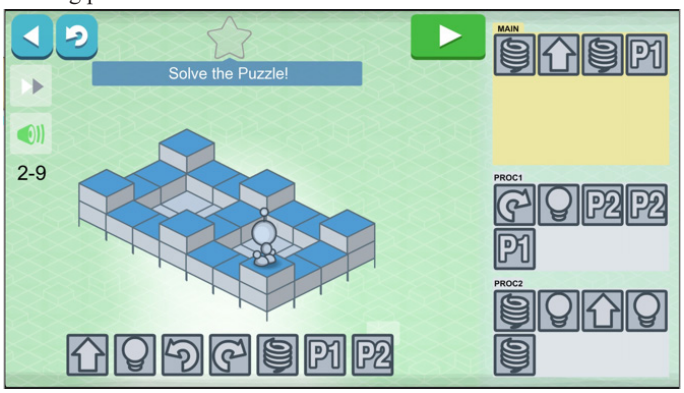

Figure 5 Level 2-9 with 14 blocks (not optimal).

We called "right, light" one and called "jump, light, forward, light, jump" two. We could also split as "right, light, jump" be one and "light, forward, light, jump" be two. There is an issue when "2" is applied twice though, getting stuck in the midpoint of the long edge. We actually need to redefine "2" to be "light, forward, light, jump, jump". This gives another valid solution, even though it needs 15 blocks. However, observe that the "jump, forward, jump" at the beginning and "light, forward, light, jump, jump" has a "least common multiple" of "light, forward, light, jump, forward, jump" which can serve both purposes. This can save two blocks and make it only 13 blocks total, which is optimal (verified by our computer code).

Figure 6 shows the result with 13 blocks.

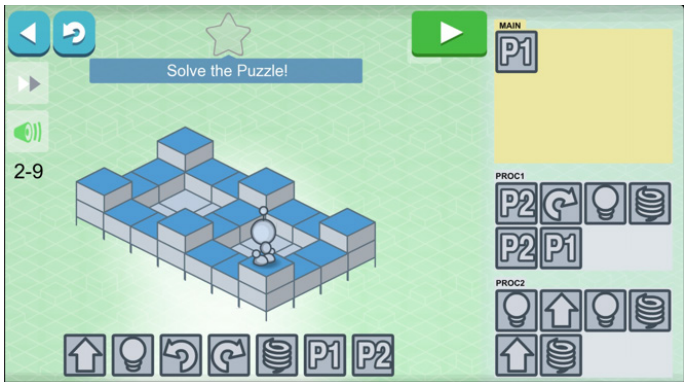

Figure 6 Level 2-9 with 13 blocks (optimal).

Level 3-2, 3-4 and 3-9

Recall that an elevator is activated by a light command when the robot is on it, and it will either go up two layers or go down four layers if it reaches the top. These three levels use the idea of cell with a shift. Observe in the map that we could either reach the same situation after a shift, or after a special start, reach the same situation with each shift. Then we could simply design a series of moves to sweep each shift, lighting up the lights within the sweep. The orientation is unchanged after the series of moves.

Figures 7 & 8 show optimal solutions for level 3-2 and 3-4. Figures 9 & 10 show the solution for level 3-9 and one middle step of it, explaining the shift. It is worth mentioning that it is quite difficult for players to come up with an optimal solution with the same number of blocks as shown here. One could have easily wasted several blocks in this level.

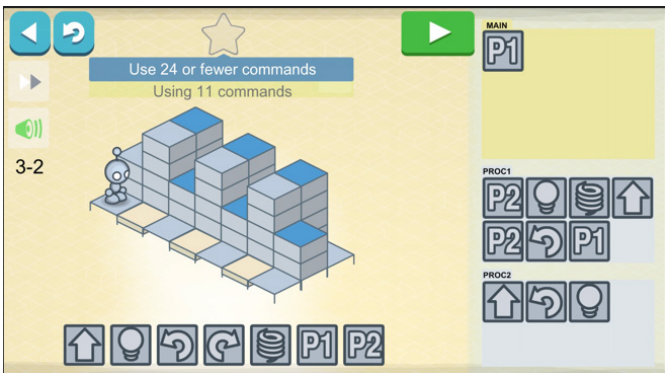

Figure 7 Level 3-2.

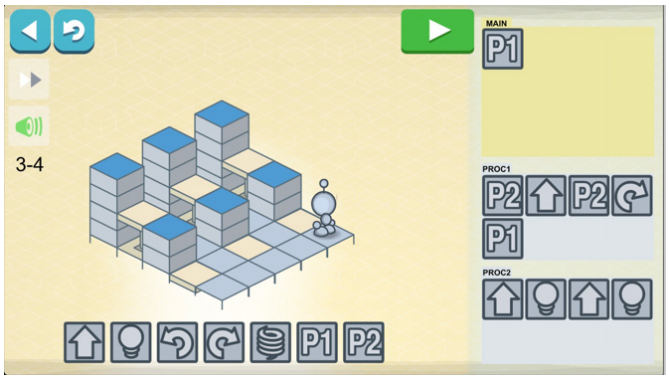

Figure 8 Level 3-4.

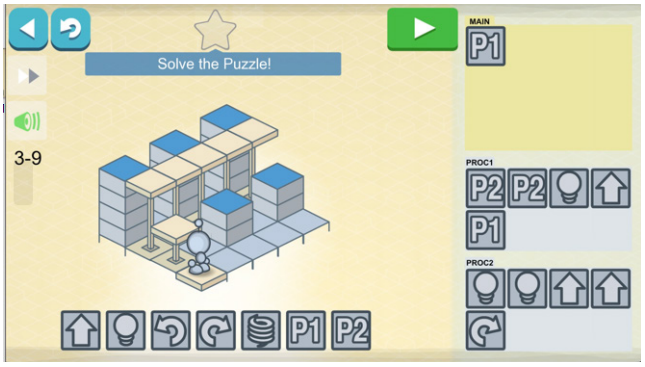

Figure 9 Level 3-9.

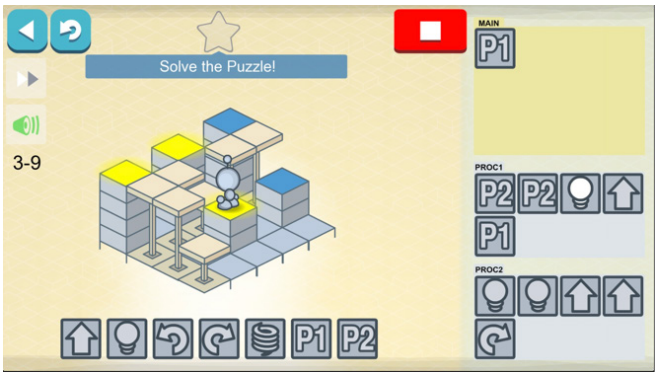

Figure 10 A middle step of Level 3-9.

Citation: Hou S, Su J. Optimal solutions for the first three stages of paid version of Lightbot. *Int Rob Auto J.* 2025;11(3):98–102.
DOI: 10.15406/iratj.2025.11.00307

Level 3-3

This uses the idea of cell with a 90 degree rotation. We have seen this in[8,9] but it was without elevators. The elevators in this level added some complications but is still manageable. Figure 11 shows the optimal solution for level 3-3. Figure 12 shows a middle step with 90 degree rotation. We can imagine that by repeating the pattern 3 more times (no need to completely finish the last repetition), all lights will be on. Note that the robot covers quite some distance by spiral moves, not just walking one round outside. However, this can indeed save block and is verified by our computer code to be optimal.

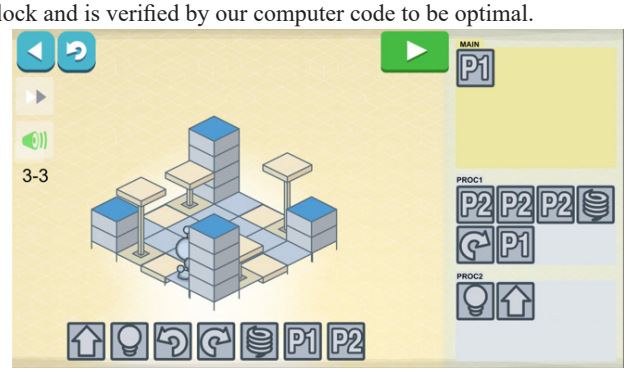

Figure 11 Level 3-3.

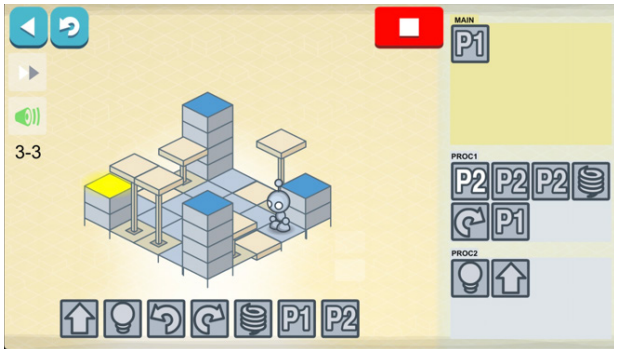

Figure 12 A middle step of Level 3-3.

Level 3-8

The green transportation spots are introduced in this level. The rule is that if the robot is on a green spot and a light command is used, then it is moved to the linked green spot with the orientation unchanged. When combined with the elevators, the level has many variations of moves and the optimal solutions are tricky. Figure 13 shows an optimal solution with only 8 blocks. The trick is after 5 periods of "jump, right, right, forward, jump, light", we use the transport to jump to a new row. Then we are in exactly the same situation, only with a shift and a 180 degree rotation. The rest follows naturally.

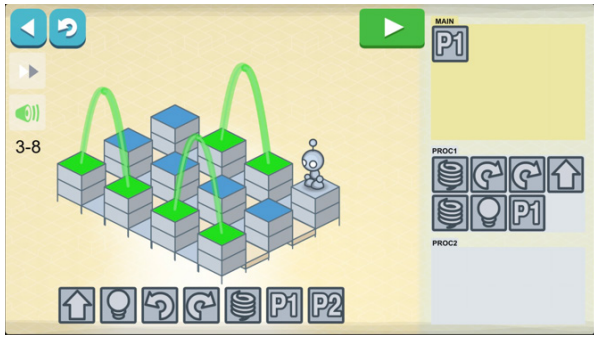

Figure 13 Level 3-8.

Conclusion

Although Lightbot is a game for kids to learn programming skills, finding optimal solutions is non-trivial especially for the paid version with complicated setup like elevators and transports. We gave detailed optimal solutions to the non-trivial levels of Lightbot (paid version) for the first three stages and a complete list of conclusions verified by our computer code for all levels of the first three stages. Some mathematical explanations were provided to reveal the beauty of symmetry. We plan to study other stages in the future.

Acknowledgment

We thank Yan Huang, Yin Sun and Sihan Yang 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

There is no conflicts of interests

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 future 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. Hou S, Su J, Huang Y. Lightbot code hour: optimal solutions, theories and designs. *International Robotics and Automation Journal*. 2025;11(1):19–23.
10. Fenyvesi C. Rubik's snake of 'infinite possibilities. The Washington Post. 1981.
11. 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.
12. 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.
13. Hou S, Chen Y, Li Z. Some mathematical problems related to the rubik's snake. *J Mech Rob*. 2021;13(1):014502.
14. Hou S, Su J, Chen Y. Palindromic, periodic and mobius¨ rubik's snakes. *International Robotics and Automation Journal*. 2021;7(3):84–88.
15. Pan E, Hou S. The legend of the rubik's snake. Lit House LLC. 2025.

Citation: Hou S, Su J. Optimal solutions for the first three stages of paid version of Lightbot. *Int Rob Auto J*. 2025;11(3):98–102.
DOI: 10.15406/iratj.2025.11.00307

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. *International Robotics and Automation Journal*. 2024;10(1):25–30.

Citation: Hou S, Su J. Optimal solutions for the first three stages of paid version of Lightbot. *Int Rob Auto J.* 2025;11(3):98–102.
DOI: 10.15406/iratj.2025.11.00307