A Comparative Analysis of DFS, Eller's and Wilson's Algorithms for Maze Generation

Jiarui Kang *

College of Arts & Sciences, Boston University, Boston, Massachusetts, United States of America

* Corresponding Author Email: kang0810@bu.edu

Abstract. Due to their theoretical importance and wide range of practical applications, maze generation algorithms have received significant attention. The generation of a perfect maze, which is defined as a structure with only one path between any two points, can be done using different algorithms that have distinct features and trade-offs. The purpose of this paper is to evaluate the performance of three widely studied methods, Deep-first Search (DFS), Eller's algorithm, and Wilson's algorithm, across time complexity, space complexity, and randomness. DFS is fast and simple, but tends to create long winding paths with structural bias. Eller's algorithm constructs mazes row by row, ensuring efficiency and memory savings while balancing connectivity and randomness. Wilson's algorithm, though slower and more memory demanding, is capable of producing uniform mazes with high randomness, making it useful for benchmarking and testing scenarios. The comparative results show that all three algorithms achieve linear time complexity but differ substantially in spatial efficiency and maze characteristics. These findings highlight the practical importance of algorithm choice: DFS is suited for real-time applications requiring speed, Eller's provides a compromise between efficiency and diversity, and Wilson's offers unbiased randomness for research or AI applications. Insights from this study are used to guide the selection of maze generation algorithms for different computational and applied contexts.

Keywords: Maze generation algorithm, DFS, Eller's algorithm, Wilson's algorithm.

1. Introduction

Maze generation algorithms are important in both the computational field and practical application. These algorithms are to construct a perfect maze which is a maze with exactly one path from any point to any other point. Maze generation algorithm plays an important role in computation, particularly in areas like randomization, problem-solving and algorithm design. mazes are also applied in psychology for studying spatial awareness and intelligence, and even in physics for modeling crystal structures [1].

Maze generation algorithms have been widely studied, and there are some famous algorithms, such as Depth-first Search (DFS), Eller's algorithm and Wilson's algorithm. DFS is a basic algorithm to explore tree structures and graph. It starts from one point and explores the rest of branches before backtracking until all the cells in the grid is explored [2]. Eller's algorithm generates a maze by progressively linking neighboring cells row by row, ensuring all nodes are connected without cycles, using a disjoint-set data structure to manage the connected components [3]. Wilson's algorithm generates a maze by performing random walks from unvisited vertices until it reaches a visited vertex, then connects the vertices along the path, ensuring no loops are created [1]. Despite their shared goal of generating perfect mazes, these algorithms differ significantly in their own features, such as time complexity, space complexity and randomness. For instance, DFS has a complicated process of generating mazes [4]. These differences are not only about theoretical perspectives, but also have practical implications. For example, the choice of an algorithm can influence the path-diversity of a maze, which is also known as randomness. In Gabrovšek's article, DFS tends to be able to create mazes with long loops and fewer intersections, which means it will generate mazes that have structural bias rather than those uniform ones [1]. In contrast, Wilson's algorithm will produce mazes that in an approximate uniform way, which have less structural bias and higher randomness [1]. Such variations of generating mazes matter when those mazes are used in some fields like artificial intelligence and gaming. For instance, simpler algorithms like Wilson's algorithm can only make

simple games, but more complex algorithms like DFS can make it possible to create complicated games [5].

This paper aims to evaluate and show these differences of the three algorithms from several perspectives, including time complexity, space complexity and randomness. By doing so, the paper can provide people with advice on choosing suitable algorithms when facing some practical problems which are essentially about generating mazes.

2. Method

This paper aim to study the differences among DFS, Eller's algorithm and Wilson's algorithm. Each algorithm's fundamental principles are widely examined. What's more, the criteria for evaluating their performance are defined. This section outlines the workings of each algorithm and will describe how they will be compared.

2.1. DFS Algorithm

DFS algorithm basically is using recursive backtracking, which will generate maze by exploring as far as possible and find one path before backtracking [1]. It starts from one random initial cell, and then the algorithm marks it as visited and chooses another cell which is marked as unvisited. This process will be repeated until all the cells around the current cell are all marked as visited, which is called "dead end". When the algorithm meets the dead end, it will backtrack to the previous cell and will check whether this cell meets a dead end too. In the end, when all the cells are marked as visited, the process will finish, and a perfect maze will be generated. DFS is so simple that tends to run very quickly due to its straightforward depth-first exploration [3]. Its principle ensures that ways are carved out in an efficient way without having any complicated data structure. In fact, DFS is the fastest algorithm for maze generation among the three algorithms which are mentioned in this article [5]. The output of DFS algorithm also has a unique feature: the mazes produced always have a main long way and have a few branches, that's because this algorithm deeply explores one way before backtracking [1]. This feature will make solving mazes generated by DFS algorithm hard, cause all the roads are twisted. However, DFS algorithm also has some cons. It is memory-intensive because this algorithm is recursive. To be more specific, recursion has the probability causing stack overflow when generating big size mazes [4].

2.2. Eller's Algorithm

Eller's algorithm generates mazes in a row-by-row approach. This algorithm maintains a data structure to keep track of connected cells in the current row. To elaborate on this idea, this algorithm starts with the top row of the grid, and assigns each cell to its own set. The walls between cells will be removed randomly to merge the sets of pairs of cells which are neighbors. Each set of the current row will carve at least one passage down into the next row to ensure that the final meze will remain connected vertically and will not have isolated regions [3]. In the next row, the cells are vertically connected in the previous step. And then they will merge randomly with cells in the same row. These two steps will repeat until the algorithm reaches the last row. In the last row, the algorithm needs to ensure that all the cells in this row are connected to different sets so that the maze becomes one single connected component. Eller's algorithm is memory-saving and time-saving. This algorithm can guarantee the connectivity without backtracking, because all the decisions are made locally, which are within each row and between rows. In addition, this can make the mazes produced by this algorithm are random and well-connected [5]. This algorithm also has some disadvantages. For example, this algorithm is more complex than others because it needs to ensure there is at least one vertical connection per set.

2.3. Wilson's Algorithm

Wilson's algorithm uses a loop-based random walk to generate mazes. This algorithm treats maze generation as a series of random walks that carve out a spanning tree in an unbiased manner [1]. To be more specific, it starts by making one cell as part of the maze, and this cell will be the seed of the growing maze tree. Then, an any other cell will be picked randomly and will be performed a random walk until reach a cell which is already in the maze tree. During this process, a loop can be formed. In addition, if the random walk ever steps into a cell she has visited in the same walk. A loop is detected and all steps in the loop are removed from the walk path. The walk then continues from the point when the loop is erased, and continue as a new cycle. This process will be repeated until all the cells have been added to the maze. The most obvious pro of this algorithm is the randomness, because its principle can guarantee the mazed generated won't have unwanted patterns [1]. But there are also some cons of this algorithm. For instance, this algorithm needs a long time to generate mazes, and it's also memory-demanding, which means it needs more memory than DFS and Eller's algorithm [4].

3. Results and Discussions

Table 1. Comparison among DFS, Eller's algorithm and Wilson's algorithm

Algorithm	Time Complexity	Space Complexity	Structural Features	Randomness
DFS	O (n), fastest	O (n)	Very long path, few intersections	low
Eller's Algorithm	O (n), efficient	O (r), based on row	Moderate intersections	moderate
Wilson's Algorithm	O (n), slowest	O (n)	Uniform intersections, many branches	high

3.1. Results Description

The results presented in Table 1 show that all three algorithms work in linear time with the respect to the number of cells. In terms of space complexity, Eller's algorithm is the most efficient algorithm, needing only the storage of one row at a time. DFS and Wilson's both need O(n) memory: DFS needs the memory for keeping the recursion stack and backtracking path; Wilson's algorithm needs to keep track on the walk path and visited states.

3.2. Result Analysis

The results show the differences of these three algorithms, and it is helpful to choose different algorithms by looking at the trade-offs through the table. DFS is suitable for applications in real life which need fast meze generation, such as the environment change of real-time game which is based on maze. However, its bias may limit the suitability for testing or AI research, because these situations need predictability. Eller's algorithm is balanced between efficiency and randomness. It is suitable for generating mazes which have large sizes due to its low memory consumption, such as mobile map generation. Eller's algorithm is not slow as Wilson's algorithm and it is memory-saving than DFS, so it can offer a compromise between DFS's efficiency and Wilson's randomness. Wilson's algorithm can generate mazes with uniform intersections, so it is valuable in those situations where randomness and unpredictability are important. For instance, Wilson's can ensure that mazes generated is uniform in AI benchmarking.

Overall, choosing the proper algorithm depends on application needs. When the speed of generation needs to be faster, DFS is the best option. When saving memory is needed, Eller's algorithm can provide a compromise. When randomness is not that necessary, Wilson's is the best option, since it is faster than Wilson's.

4. Conclusion

In this work, the efficiency, structural features and randomness of DFS, Eller's algorithm and Wilson's algorithm are compared. The results shows that their trade-offs are important. In conclude, DFS needs the shortest time to generate mazes and generate simple mazes; Eller's algorithm gives a balance between generating time and randomness; Wilson's algorithm needs the longest time and generates the most complex mazes. However, there are some limitations of this paper. First, there are only three algorithms are compared, but other algorithms may offer better trade-offs too. Also, the evaluation of this paper is mainly based on the published results instead of experiences, which may decrease the accuracy. A way to improve this is to evaluating these algorithms in a quantified experience, which has a uniform standard to help compare these algorithms.

Looking ahead, further research could expand the scope of comparison to include other algorithms, such as Kruskal's algorithm, Prim's algorithm, or hybrid approaches that combine randomization with heuristic guidance. Such extensions would facilitate a broader understanding of the trade-off between efficiency and complexity in diverse scenarios.

Furthermore, large-scale experimental evaluations under controlled and reproducible conditions would facilitate the establishment of reliable maze generation benchmarks. These benchmarks could cover multiple evaluation dimensions, such as generation time, memory usage, structural diversity, and practical usability in applications such as game design, robotics simulation, and procedural content creation. Another promising direction is the exploration of adaptive or dynamic maze generation algorithms that adjust their strategies based on specific user-defined requirements, such as balancing challenge and solvability in real time.

Furthermore, incorporating visualization and user study components could provide insights into how different algorithms affect human player perceived difficulty and engagement, linking computational results to experiential evaluations. Ultimately, systematic, quantitative, and user-centered approaches will not only improve the evaluation of maze generation algorithms but also ensure their applicability in both academic and real-world settings.

References

[1] Gabrovšek P. Analysis of maze generating algorithms. In: Proceedings of ICUSI 2024. p. 2 – 3.

[2] Elkari B, Ourabah L, Sekkat H, Hsaine A, Essaïouad C, Bouargane Y, El Moutaouakil K. Exploring maze navigation: a comparative study of DFS, BFS, and A* search algorithms. Stat Optim Inf Comput. 2024; 12 (3): 761 – 8. doi: 10.19139/soic-2310 - 5070 - 1939.

[3] Pasca M, Nandra C. Design and implementation of a 2D game engine: algorithmic approaches and performance optimization. In: Proceedings of ICUSI 2024. p. 74 – 5.

[4] Karlsson A. Evaluation of the complexity of procedurally generated maze algorithms [bachelor's thesis]. Blekinge: Blekinge Institute of Technology; 2018.

[5] Čarapina M, Staničić O, Dodig I, Cafuta D. A comparative study of maze generation algorithms in a game-based mobile learning application for learning basic programming concepts. Algorithms. 2024; 17 (9): 404. doi: 10.3390/a17090404.