Jurnal Informatika dan Rekayasa Perangkat Lunak ISSN 2656-2855
Vol. 8, No. 1, March 2026, Page. 90-101 e-ISSN 2685-5518

Comparison of Blind Search and Heuristic Search Algorithms in Finding Solutions to the 8-Puzzle Game: Introduction to Endemic Wallacea Animals

Mutmainnah Muchtar[1*]
[1*] Department of Computer Science, Faculty of Information Technology,
Universitas Sembilanbelas November Kolaka
[*]Email: muchtarmutmainnah@gmail.com

Abstract

Search algorithms play a crucial role in artificial intelligence, particularly in solving pathfinding and combination problems such as the 8-Puzzle game. This study presents the development of a web-based 8-Puzzle game designed to introduce endemic Wallacea fauna by comparing the performance of Blind Search and Heuristic Search algorithms. The system is built using HTML, CSS, and JavaScript, developed with Notepad++, and executed using a standard web browser. Four search algorithms are implemented, consisting of two blind search methods (Breadth First Search and Depth First Search) and two heuristic search methods (Greedy Best First Search and A). Performance testing is conducted using three scenarios: testing easy puzzle configurations, testing high-complexity configurations, and cross-platform testing on desktop and mobile devices. The experimental results show that heuristic search consistently outperforms blind search. A* produces optimal solution paths with fewer expanded nodes, while Greedy achieves the fastest execution time. In contrast, DFS performs the worst, requiring in-depth node exploration and long execution time. Multi-platform evaluation shows that the game runs quite smoothly on both desktop and mobile devices. These results indicate that heuristic search, especially A*, is the most effective method for solving the educational 8-puzzle game of Wallacea endemic animal recognition.*

***Keywords**: 8-Puzzle, Blind Search, Endemic, Heuristic Search, Wallacea Animals*

INTRODUCTION

The Wallacea region is the central part of Indonesia with an area of 347,000 km^2, namely Sulawesi Island, part of Nusa Tenggara, Halmahera and also including the country of Timor Leste (Bisjoe, 2015). Its name comes from the surname of the English naturalist and explorer: Alfred Russel Wallace. This area is rich in biodiversity, as each island has endemic flora, fauna, and microorganisms. (Mayasari et al., 2018; Wulandari et al., 2023).

Wallacea's endemic fauna refers to unique species found only in the Wallacea biogeographic region. This region has a high level of endemism, so many species, such as the Maleo (Macrocephalon maleo), Tarsius (Tarsius spectrum), Babirusa (Babyrousa celebensis), Bear Cuscus (Ailurops ursinus), and Komodo dragon (Varanus komodoensis) are found nowhere else on Earth (Nasruddin et al., 2023). Wallacea acts as an evolutionary bridge, shaped by millions of years of geographic isolation that have resulted in unique adaptations and biodiversity. Understanding Wallacea's endemic wildlife is crucial because many of these species are classified as threatened or endangered due to habitat loss, deforestation, and the illegal wildlife trade (Maskun et al., 2020; Pranoto et al., 2015). Raising public awareness, especially through education and interactive media such as educational games, can support conservation efforts and emphasize the importance of preserving this irreplaceable species for future generations (Ani, 2022).

Advances in information technology have opened up significant opportunities for the development of interactive and educational digital learning media (Rustan et al., 2022; Sari et al., 2024). One effective approach is game based learning, which utilizes digital games as a learning medium to increase user motivation and understanding of a topic. This model has proven effective in combining entertainment and education, making the learning process more enjoyable and meaningful (Coleman & Money, 2020).

AI (Artificial Intelligence) is playing an increasingly important role in education by enabling adaptive and interactive learning experiences that enhance student understanding and engagement (Rifky et al., 2024). The 8-Puzzle game is a classic example often used to introduce the concepts of search algorithms and state space representation (Ando & Takefuji, 2020). This game challenges users to move eight numbered or pictured tiles in a 3×3 grid until they achieve the correct configuration. Solving the 8-Puzzle requires systematic exploration of the solution steps, making it suitable for both Blind Search and Heuristic Search algorithms.

Results of several provious studies indicate that both blind and heuristic search methods can provide good performance, especially heuristic algorithms that use estimation functions, making them more efficient in large search spaces. However, there is still little research comparing several blind and heuristic search algorithms simultaneously on an 8-puzzle game with various configurations. By comparing blind and heuristic search algorithms in an 8-puzzle game for recognizing endemic animals in Wallacea, this study is expected to provide a deeper understanding of the performance of each method. In addition, the results of this study can also support the development of interactive learning systems that are not only educational but also introduce the richness of endemic animals in Wallacea.

LITERATURE REVIEW

2.1 Previous Study

Several previous studies have utilized blind search algorithms and heuristics to solve game solution finding problems. For example, research from (Ivanochko et al., 2025) optimized the solution of 15-Puzzle using Breadth First Search and Depth First Search algorithms, and found that BFS is more efficient in finding optimal solutions, while DFS is faster in deep search conditions but not always optimal. Other studies by (Safrizal & Rozak, 2019) and (Wijayanti et al., 2020) also applied heuristic approaches such as Greedy Best First Search and A* to improve efficiency in Android-based educational games. These results show that both non-heuristic and heuristic-based approaches can provide good performance. This is especially true for heuristic algorithms, as they use estimation functions to assess the distance to a solution, making them more efficient in large search spaces. (Muchtar & Wellem, 2025)actually developed an educational game puzzle to recognize Sulawesi's endemic animals using the BFS Algorithm. However, there are not many studies comparing several blind search algorithms and heuristics simultaneously in a case study of an 8-puzzle game with various configurations.

Based on several previous studies, it can be concluded that both blind and heuristic search algorithms have their respective advantages in solving puzzle games. Although various methods have been applied to educational games and puzzles, research comparing these algorithms simultaneously in the 8-puzzle case is still limited. Therefore, this study is expected to contribute to analyzing the performance of various search algorithms and support the development of a more effective educational game for recognizing endemic animals of Wallacea.

2.2 8-Puzzle Game

8-Puzzle is a puzzle game consisting of numbered or illustrated squares that must be arranged in the correct order. It consists of 8 squares and 1 empty space that can be moved up, down, right, or left. The puzzle is 3×3, so the squares can only move within that limit. Once randomized, players must find a way to reassemble the puzzle. This game also serves as an excellent model for learning search algorithms and problem-solving strategies. As an educational game, it helps develop problem-solving skills, logical thinking, and an understanding of search algorithms (Muchtar & Wellem, 2025). When combined with specific thematic content such as images of endemic animals, this game can also be a tool to increase knowledge or awareness in a fun and interactive way.

Figure 1. Example of number and image version of 8-puzzle

2.3 Blind Search

Blind Search, also known as Uninformed Search, refers to a search strategy that explores a problem space without any additional knowledge about the goal location. These algorithms rely solely on the problem definition: the initial state, possible actions, and goal trials to systematically explore all possible paths. Because they don't evaluate which path is most promising, blind search tends to expand more nodes and require longer computational time as the search space grows. Two basic algorithms in this category are Breadth First Search (BFS) and Depth First Search (DFS).

BFS search is a level-order search algorithm that expands all nodes to a certain depth before moving to the next node (Muchtar & Wellem, 2025). This algorithm uses a queue structure (First In First Out) to store unexplored vertices. The evaluation function for BFS is defined as follows:

$$f(n) = g(n) \quad (1)$$

where *g(n)* represents the cost or number of steps from the starting node to the current node *n*.

BFS is complete and optimal, always finding the shortest path. However, its drawback is its high memory usage, as it stores all nodes at each expansion level. In contrast, DFS explores nodes by going as deep as possible along a single branch before backtracking. DFS uses a stack structure (Last In First Out) to manage the search process (Wicaksono & Trisnawan, 2021). The search process can be represented as:

$$f(n) = depth(n) \quad (2)$$

DFS is memory efficient and often faster on deep search trees, but it does not guarantee finding the shortest path. DFS can also experience infinite loops if the search space is not well constrained or managed. Therefore, in general, BFS is better when the shortest solution is required, while DFS is suitable when memory resources are limited or the depth of the solution is unknown.

2.4 Heuristic Search

Heuristic Search, also known as Informed Search, is a search strategy that uses additional knowledge or heuristic information to guide the search process towards a goal more efficiently (Abraham et al., 2015). Unlike blind search, heuristic search evaluates which nodes are most promising to explore next by estimating the distance or cost from the current state to the goal state. Heuristic search aims to reduce the number of nodes explored and the execution time by prioritizing paths that appear more optimal.

GBFS (Greedy Best First Search) selects the next node based solely on the heuristic value *h(n)* (Wijayanti et al., 2020). This search always expands the node that appears closest to the goal according to the heuristic. Below is the formula of Greedy Best First Search:

$$f(n) = h(n) \quad (3)$$

GBFS is fast and memory efficient because it focuses solely on estimating proximity to the goal. However, GBFS does not guarantee the shortest solution and may get stuck in local optima if its heuristics are not accurate.

Meanwhile, A* combines the benefits of BFS (finding the shortest path) and GBFS (reducing search effort). A* evaluates each node based on its total estimated cost with the following formula:

$$f(n) = g(n) + h(n) \quad (4)$$

where:
g(n) = cost from start to node *n*
h(n) = heuristic estimate from node *n* to goal

A* is complete and optimal because its heuristic or approximation function is always optimistic or accurate, and never overestimates it. This ensures that A* does not miss optimal paths by initially ignoring paths that appear more expensive (Safrizal & Rozak, 2019). In many cases, A* expands fewer nodes than BFS and produces better solutions than Greedy.

2.5 Wallacea's Endemic Wildlife

The Wallacea region, located between the Asian and Australian biogeographic zones, encompasses some of the world's most unique endemic wildlife. This transitional region encompasses Sulawesi, Maluku, Nusa Tenggara, and several surrounding islands, where millions of years of geographic isolation have created distinct evolutionary pathways for flora and

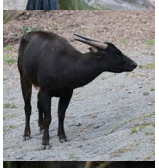
fauna. Wallacea is home to rare birds such as the Maleo (*Macrocephalon maleo*), known for burying its eggs in geothermally heated sand; the Yellow-crested Cockatoo (*Cacatua sulphurea*), a parrot endangered by the illegal wildlife trade; and the Halmahera Angelbird (*Semioptera wallacii*), a bird of paradise found only in the forests of Halmahera. Iconic mammals include the Anoa (*Bubalus depressicornis*), a dwarf buffalo native to Sulawesi's dense tropical forests; the Babirusa (*Babyrousa celebensis*), recognizable by its upward curving tusks; and the nocturnal Tarsier (*Tarsius spectrum*), one of the world's smallest primates. This area is also home to unique reptiles such as the Komodo dragon (*Varanus komodoensis*), the largest lizard on Earth. These species demonstrate the Wallacea region as a center of biologically unique yet highly vulnerable biodiversity, necessitating conservation and public awareness (Struebig et al., 2022)

METHODS

3.1 Dataset Collection

The dataset used in this study includes 15 images of Wallacea's endemic fauna, consisting of 5 bird species, 5 mammal species, and 5 reptile species. Examples of these animals include the burung Maleo, burung Elang Flores, Anoa, Kuskus beruang, Komodo, Kadal Terbang Sulawesi and others. Each image is accompanied by a short descriptive text explaining its habitat, behavior, and conservation status. The images are used as the background of a puzzle, which is then divided into 3×3 tiles. All images were obtained from freely available websites. The size of each sample image was then equalized. The images are stored in a folder named `/img` and are dynamically loaded into the puzzle board during execution. Table 1 shows the 15 endemic Wallacea animal image data used in this study.

Table 1. Wallacea endemic animal image data

No	Fauna's Name	Image on Puzzle
1	Maleo	
2	Kakatua Jambul Kuning	
3	Bidadari Halmahera	
4	Elang Flores	
5	Kangkareng Sulawesi	
6	Anoa	
7	Babirusa	
8	Kuskus Beruang	
9	Tarsius	
10	Bajing Kerdil Sulawesi	
11	Komodo	
12	Kadal Terbang Sulawesi	
13	Kura-kura Leher Ular Pulau Rote	

No	Fauna's Name	Image on Puzzle
14	Ular Mangrove Sulawesi	
15	Soa Layar	

Source : *google image* and wikipedia

3.2 System Design

This system was developed using HTML, CSS, and JavaScript as a web-based interactive educational game. The design process followed the Waterfall approach (Muchtar & Wellem, 2025) with the following steps:

1. Requirements analysis: defining puzzle mechanics and algorithm integration.
2. Interface design: designing the puzzle layout, drag-style puzzles, image slicing, and modal information displays.
3. Algorithm module design: creating data structures that represent puzzle states, transitions, and goal checks.
4. Testing: measuring the performance of each of the Blind Search and Heuristic Search algorithms.

The architecture consists of:

1. Front-end interface: displays the tiles of the 8-puzzle and buttons (Shuffle, Solution using A*, and Reset).
2. Logic layer: represents the state of the 8-puzzle using arrays and handles tile movement.
3. Search algorithm module: runs BFS, DFS, Greedy, or A* behind the scenes to compare performance.

3.3 Algorithm Implementation

In this stage, four types of search algorithms are implemented to solve the 8-puzzle problem of endemic animals of Wallacea. Table 2 shows the four algorithms that will be implemented in this study. All algorithms manipulate the state of the puzzle using node expansion and state validation functions to avoid state repetition. The `performance.now()` function in JavaScript is used to measure execution time precisely in milliseconds. After the puzzle is solved, the results are displayed via a pop-up menu containing a performance comparison between the four algorithms.

Table 2. Four types of algorithms are used

Algorithm	Type	*Cost function*
BFS	Blind Search	Explores nodes level by level
DFS	Blind Search	Explore the deepest branches first
Greedy	Heuristic Search	$f(n) = h(n)$
A*	Heuristic Search	$f(n) = g(n)+h(n)$

3.4 Performance Testing and Measurement

Performance testing was conducted to evaluate and compare the efficiency of BFS, DFS, Greedy Best First Search, and A* in solving the 8-Puzzle. The following test scenarios were performed:

1. Scenario 1: Complexity Test (Easy and Medium Levels).
   In this scenario, performance was tested on several randomized puzzle configurations at easy and medium levels. Each algorithm was executed on the same configuration, and three performance metrics were measured: the number of solution steps, the number of expanded nodes, and execution time. This scenario identifies how efficiently each algorithm performs at standard puzzle difficulty levels.
2. Scenario 2: Complexity Test (Hard Level).
   The puzzle configurations were intensively randomized to create a difficult search space. This scenario demonstrates the algorithm's robustness when the search space becomes large and the branching increases significantly.
3. Scenario 3: Multiplatform Test.
   The system was tested on various devices (desktop/laptop and mobile) and various browsers. This test checks whether the puzzle loads correctly, the buttons function (randomize, find solution, reset), the animations run smoothly, and the algorithm execution time remains acceptable. The goal is to ensure that the web based game can be used across multiple platforms.

RESULT AND DISCUSSION

To compare the performance of blind and heuristic search methods, a web-based interface was developed for the 8-Puzzle educational game, which introduces Wallacea's endemic fauna. The game uses HTML, CSS, and JavaScript, designed and edited with Notepad++ as the primary text editor, and then executed through a standard web browser (Google Chrome, Mozilla Firefox, or Microsoft Edge). The system runs on a 64-bit Windows 11 operating system with 16 GB of RAM and an AMD Ryzen 7000 processor.

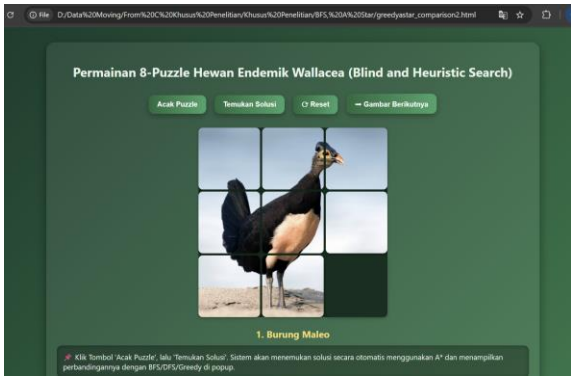


Figure 2. Main user interface

The system architecture is built following the Waterfall development model, which consists of sequential stages: requirements analysis, design, implementation, testing, and evaluation. This model was chosen because it provides a structured approach suitable for small to medium-scale educational software projects. Figure 3 shows the interface of the web-based 8-puzzle game that has been built and run with the Google Chrome browser. The language used is Indonesian. Meanwhile, the following code is used to implement the board/tile creation part of the 8-puzzle game using the `buildBoard()` function. This code will select the active endemic animal image, then divide it into tiles that are displayed on the puzzle grid. Each tile has a background position according to the original image piece, and if the tile is not an empty area, it can be clicked to be moved.

```
function buildBoard(){
  const
img=animals[currentAnimal].file;
  board.innerHTML="";
  for(let i=0;i<N*N;i++){
    const v=state[i];
    const
tile=document.createElement("div");
    tile.className="tile"+(v===0?"
blank":"");
    const
inner=document.createElement("div");
    inner.className="tile-inner";
    if(v!==0){
inner.style.backgroundImage='url(${img
})';
inner.style.backgroundPosition=backgro
undPosForValue(v); }
    tile.appendChild(inner);
    if(v!==0)
tile.onclick=()=>tryMove(i);
    board.appendChild(tile);
  }
nameEl.textContent=animals[currentAnim
al].name;
}
```

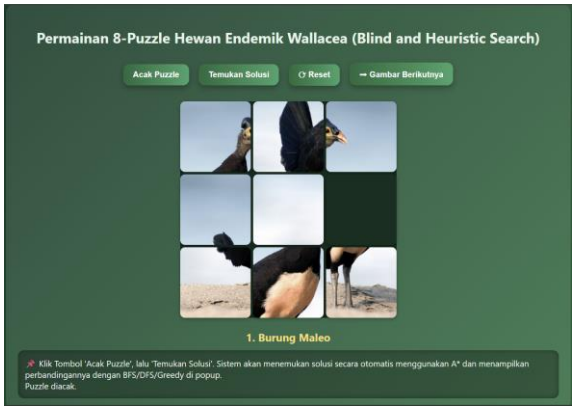


Figure 3. View when shuffling the puzzle

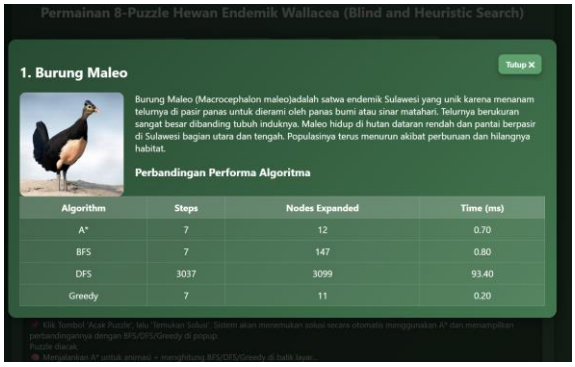


Figure 4. Pop-up menu display

Figure 3 shows the system interface for the puzzle randomization section when the user presses the “Acak Puzzle” button. Meanwhile, Figure 4 shows the system interface display when the user successfully solves the randomized puzzle, or when the user presses the “Temukan Solusi” button. This section contains the name of the endemic animal, brief

information or profile of the animal, and a performance comparison of four types of search algorithms. This pop-up menu can be closed by pressing the "Tutup" button and will return to the main menu. The following is the code implementation for randomizing the puzzle. The `shuffle()` function functions to randomize the position of the tiles in the puzzle so that the game starts in a random state. This code randomly swaps the position of the blank tile with its neighboring tile 11 times. After the randomization process is complete, the board display is updated via `buildBoard()` and the message "Puzzle diacak." is displayed in the game log.

```
function shuffle(){
  for(let i=0;i<11;i++){
    const b=state.indexOf(0);
    const n=neighborsIdx(b);
    const
p=n[Math.floor(Math.random()*n.length)];

[state[b],state[p]]=[state[p],state[b]];
  }
  buildBoard(); log("Puzzle diacak.");
}
```

Next, a test was conducted designed to compare the performance of blind search and heuristic search algorithms consisting of four search algorithms, namely BFS, DFS, Greedy Best First Search, and A-Star, in solving the 8 Puzzle problem for the Wallacea Endemic Animal Recognition Game. The purpose of this evaluation is to analyze the efficiency of the method in terms of computation of each algorithm when applied to the same puzzle configuration. The testing process measures three main metrics, namely the number of steps required to reach the goal state, the number of nodes expanded during the search, and the execution time in milliseconds.

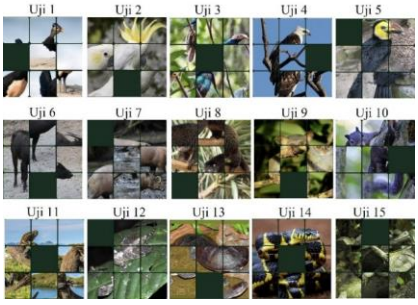


Figure 5. Randomized puzzle configuration for first test

Table 3. Low to middle complexity performance test results

Testing		BFS	DFS	Greedy	A*
Uji 1	Steps	7	55	7	7
	Node	129	57	12	18
	Time (ms)	0.70	0.30	0.10	0.80
Uji 2	Steps	5	835	5	5
	Node	41	852	7	8
	Time (ms)	0.20	9.40	0.10	0.00
Uji 3	Steps	8	464	8	8
	Node	244	474	9	9
	Time (ms)	1.50	4.20	0.10	0.70
Uji 4	Steps	3	899	3	3
	Node	17	915	4	4
	Time (ms)	0.00	8.40	0.00	0.10
Uji 5	Steps	4	430	4	4
	Node	17	438	5	5
	Time (ms)	0.30	5.70	0.20	0.40
Uji 6	Steps	5	2193	5	5
	Node	36	2241	6	6
	Time (ms)	0.20	59.60	0.20	0.20
Uji 7	Steps	6	434	6	6
	Node	57	444	7	7
	Time (ms)	0.70	6.80	0.30	0.90
Uji 8	Steps	8	2598	8	8
	Node	173	2652	272	23
	Time (ms)	1.10	66.40	6.80	0.90
Uji 9	Steps	3	829	3	3
	Node	15	846	4	4
	Time (ms)	0.30	15.60	0.20	0.60
Uji 10	Steps	7	427	7	7
	Node	116	435	8	8
	Time (ms)	0.30	2.80	0.10	0.10
Uji 11	Steps	8	8	32	8
	Node	337	9	193	27
	Time (ms)	1.50	0.00	2.60	0.90
Uji 12	Steps	4	430	4	4
	Node	17	438	5	5
	Time (ms)	0.00	2.90	0.00	0.10
Uji 13	Steps	5	25	5	5
	Node	53	26	6	6
	Time (ms)	0.20	0.20	0.00	0.10
Uji 14	Steps	2	434	2	2
	Node	9	444	3	3
	Time (ms)	0.00	3.31	0.10	0.00
Uji 15	Steps	7	433	7	7
	Node	153	444	8	8
	Time (ms)	1.80	6.20	0.40	1.20

Figure 5 shows the 15 puzzle configurations used in the first test scenario. In this scenario, the puzzles were randomized to low to medium difficulty levels, as they represent the conditions a typical user might encounter during routine use of the game. The experimental results presented in Table 3 show a baseline performance comparison between the Blind Search algorithms (BFS and DFS) and the Heuristic Search algorithms (Greedy Best First Search and A*) under easy or normal game conditions. For each trial, three performance

metrics were measured: (1) the number of solution steps, which indicates the length of the final solution path; (2) the number of nodes expanded, which reflects how much of the search space the algorithm had to explore; and (3) the execution time, measured in milliseconds, which indicates how quickly the algorithm produced a solution.

From these first tests, BFS and A* generally produced solutions with the fewest steps. This means they tend to reach their goal using a sequence of steps that is close to optimal. Greedy Best First Search was consistently the fastest in terms of execution time, often solving the puzzle in a fraction of a millisecond, while also expanding only a few nodes. However, Greedy does not guarantee the shortest solution path available, and in some cases its number of steps and nodes was slightly higher than A* (Test 8 and Test 11). Meanwhile, DFS performed the worst overall in this scenario; while sometimes completing quickly on simple randomization, it often explored many unnecessary states, resulting in very high node expansion and, in some trials, significantly longer execution times compared to the other algorithms. Thus, in low- or normal-difficulty puzzles, A* showed the best balance between path optimality and reliability, Greedy was the fastest, BFS was acceptable but less scalable, and DFS was the least efficient in terms of search.

These results confirm that heuristic-based algorithms (A* and Greedy Best First Search) are significantly more effective and more scalable for solving the 8-Puzzle game of Wallacea endemic animal images for low or normal difficulty levels, while blind search methods are slower and require expensive computation.

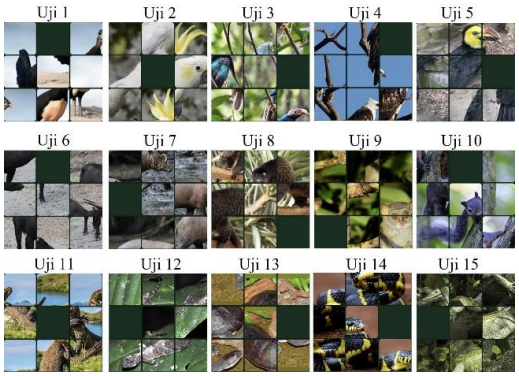


Figure 6. Randomized puzzle configuration for second test

Table 4. High complexity performance test results

Testing		BFS	DFS	Greedy	A*
Uji 1	Steps	17	16895	45	17
	Node	16894	17309	1156	993
	Time (ms)	27.90	1306.5	15.00	15.20
Uji 2	Steps	12	3816	24	12
	Node	1502	3900	567	138
	Time (ms)	3.70	71.00	5.60	1.10
Uji 3	Steps	21	19501	45	21
	Node	58624	19985	691	6110
	Time (ms)	79.10	1733.7	6.10	449.6
Uji 4	Steps	24	11702	58	24
	Node	129767	11967	806	18556
	Time (ms)	2346.30	567.70	8.10	4308.6
Uji 5	Steps	15	43	19	15
	Node	8455	45	83	385
	Time (ms)	16.0	0.20	0.50	4.50
Uji 6	Steps	17	7385	31	17
	Node	19552	7545	344	776
	Time (ms)	69.5	230.4	3.10	9.70
Uji 7	Steps	21	1605	47	21
	Node	69324	1638	1173	5069
	Time (ms)	86.7	19.3	15.1	271.5
Uji 8	Steps	20	9952	34	20
	Node	44683	10173	217	3568
	Time (ms)	59.70	217	1.30	143.0
Uji 9	Steps	20	22772	74	20
	Node	50324	23331	2088	2886
	Time (ms)	83.3	2784.5	42.9	94.6
Uji 10	Steps	19	14551	49	19
	Node	42045	14894	839	2146
	Time (ms)	61.6	916.2	8.80	53.50
Uji 11	Steps	22	21886	58	22
	Node	90744	22419	639	9746
	Time (ms)	1220.4	2132.0	5.80	1144.9
Uji 12	Steps	23	5567	73	23
	Node	116056	5688	1507	14379
	Time (ms)	1181.2	107.1	22.8	2701.9
Uji 13	Steps	17	24691	25	17
	Node	16161	25319	172	915
	Time (ms)	27.2	2689.0	0.90	13.8
Uji 14	Steps	23	15013	45	23
	Node	100051	15377	816	14065
	Time (ms)	212.0	992.0	9.4	2532.0
Uji 15	Steps	19	6727	81	19
	Node	33050	6873	972	2196
	Time (ms)	49.8	180.3	12.30	56.10

The second test scenario, called the complexity test, was designed to test the algorithm's scalability under increasingly difficult puzzles. In this scenario, the 8-Puzzle game was intentionally randomized to generate configurations at a difficult level (highly random but still solvable). Figure 6 shows the 15 puzzle configurations used in this second test scenario. Similar to the first test scenario, the same three metrics were used: the number of steps to the solution, the number of vertices generated, and the execution time. However, the goal here was

not simply to see whether the algorithm could solve the puzzle, but rather to see how its computational cost increased as the required solution depth increased (e.g., puzzles requiring 17, 20, or 24 moves).

Table 4 shows the results of this complexity test. Under these more challenging conditions, the differences between the algorithms become even more pronounced. A* consistently matches BFS in terms of optimality (both often achieving the minimum number of steps required to reach the goal), but it does so with significantly fewer node expansions than BFS. For example, when the required solution length is 20+ steps, BFS sometimes expands tens of thousands to over a hundred thousand nodes, while A* solves the same puzzle with significantly fewer expansions. However, A* can exhibit higher runtime costs in very complex puzzles, for example, in very difficult cases, its execution time increases to hundreds or even thousands of milliseconds due to the multiple evaluations and prioritizations.

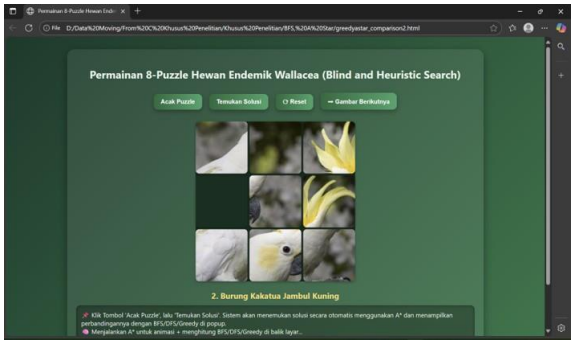


Figure 7. Test results on the desktop platform with the Microsoft Edge browser

Greedy search yields mixed results. It is typically the fastest in terms of execution time, even on difficult puzzles, and tends to expand very few nodes compared to BFS and DFS. However, in these high-difficulty tests, Greedy does not always produce the shortest solution path. Unlike A* and BFS, which consistently produce the shortest number of steps, Greedy consistently produces a higher number of steps than A* and BFS, despite remaining the fastest in terms of time. This confirms that Greedy sacrifices path optimality for speed. Meanwhile, DFS is the least reliable and least scalable. In cases of high complexity, DFS expands a very large number of nodes and sometimes requires very long search times (up to a second or more), or produces very long solution paths, due to the need for unsupervised (non-heuristic) exploration.

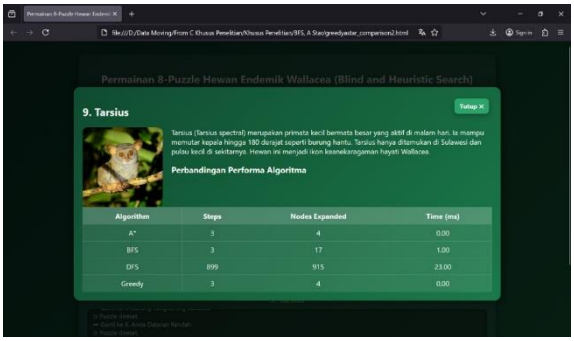


Figure 8. Test results on desktop platforms and Mozilla Firefox browser

The third test scenario was conducted to evaluate the multi-platform performance of this web based 8-puzzle game on different devices, specifically between desktop and mobile browsers. The primary goal of this test was to determine whether each puzzle algorithm and animation could run smoothly and remain responsive on both platforms, given that the system was designed as a web browser based educational tool.

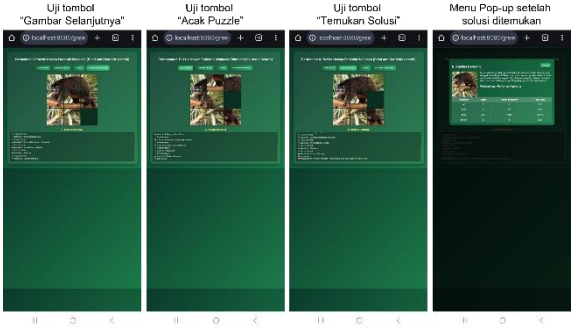


Figure 9. Test results on mobile platform with chrome browser

For desktop testing, the game was run on a laptop with a standard Windows 11 operating system (16GB RAM, AMD Ryzen 7000 series processor) using Mozilla Firefox and Microsoft Edge. On a mobile smartphone, the game was run on an Android OS (Samsung Galaxy A14 5G, Octa-core CPU with 6GB RAM) using the Google Chrome browser. During testing, the smoothness of animation transitions and game execution time were recorded.

Table 5. Cross-Platform Testing Results

Test Parameters	Description	Desktop	Mobile
Loading time:	The time it takes for the game interface (images, buttons, layout) to fully load.	Less than 1 second	1-2 seconds
Animation smoothness	Responsiveness of the "Acak" button and tile arrangement animation.	Very smooth and stable	Smooth, with minor lag
Average execution time	The average time it takes the heuristic algorithm to find a solution.	0.79 seconds	2.4 seconds
Response when a button is clicked	Delay in reaction after the user clicks or moves a tile or puzzle button.	Immediate response	<0.2 seconds
Cross-device consistency	Algorithm results (same number of steps, nodes, and time across devices).	Identical	Identical

Figures 7 and 8 depict the game interface when run on a desktop device using Microsoft Edge and Mozilla Firefox browsers. Meanwhile, Figure 9 shows the game interface when run on a mobile device or smartphone using the Google Chrome browser. Table 5 shows the results of this cross-device testing. The results indicate that the desktop platform consistently achieves faster execution times due to its greater processing power and memory compared to mobile. However, mobile browsers are still able to execute both the blind search and heuristic algorithms efficiently, with only slight differences in animation speed and smoothness. These results demonstrate that the web-based implementation of this 8-puzzle game is device independent and performs quite reliably on both laptops and smartphones. This cross-platform capability enhances the educational value of the system, as it is easily accessible and engaging for anyone wishing to access it on both desktop and mobile devices.

CONCLUSION

The Greedy Best First Search algorithm is able to achieve the shortest execution time with the fewest nodes, while the A* algorithm consistently produces optimal solution paths with fewer generated nodes than the Breadth First Search algorithm. This shows that the Heuristic Search-based model is more efficient than the Blind Search. The Breadth First Search algorithm can still find the optimal solution, but it requires exploring a large number of nodes, making it computationally expensive. Meanwhile, Depth First Search has the worst performance, often generating thousands of nodes and taking the longest to complete. This suggests that DFS is less suitable for the 8-Puzzle game, which has a deeper state space. Performance testing on desktop and mobile platforms and three different browsers showed that the A* algorithm, the primary solution-finding method in this game, had an average execution time of 0.79 seconds on desktop and 2.4 seconds on mobile. This demonstrates that the game is responsive and compatible on both devices.

Future works could develop the game with more complex puzzle configurations and compare more search algorithms and artificial intelligence. Furthermore, interactive educational features and evaluation of user learning effectiveness could be added to improve the educational game for recognizing endemic animals of Wallacea.

REFERENCES

Abraham, D., Permana, I. W., Adi Nugraha, R., Alvian, M., Teknik Elektro, J., & Sultan Ageng Tirtayasa Cilegon, U. (2015). Penyelesaian Masalah 8-Puzzle dengan Algoritma Steepest-Ascent Hill Climbing. *SETRUM*, *4*(1), 40–44.

Ando, R., & Takefuji, Y. (2020). *A new perspective of paramodulation complexity by solving massive 8 puzzles*. http://arxiv.org/abs/2012.08231

Ani, A. F. (2022). Penggunaan Media Wayang Tumbuhan dan Hewan Untuk Meningkatkan Hasil Belajar IPS Materi

Persebaran Flora dan Fauna di Indonesia Siswa Kelas Inklusi VII D Tahun Pembelajaran 2022/2023 Di SMP Negeri 2 Bontang. *Jurnal Ilmiah Global Education*, *2*(3), 144–154.

Bisjoe, A. R. H. (2015). Kawasan Wallacea dan Implikasinya Bagi Penelitian Integratif Lingkungan Hidup dan Kehutanan. *Info Teknis EBONI*, *12*, 141–148.

Coleman, T. E., & Money, A. G. (2020). Student-centred digital game–based learning: a conceptual framework and survey of the state of the art. *Higher Education*, *79*(3), 415–457. https://doi.org/10.1007/s10734-019-00417-0

Ivanochko, I., GregušMl, M., & Treiová, S. (2025). Optimizing the 15 Puzzle with AI: Comparative analysis of breadth-first and depth-first search algorithms. *Procedia Computer Science*, *265*, 57–64. https://doi.org/10.1016/j.procs.2025.07.156

Maskun, Naswar, Achmad, Assidiq, H., & Raisman, J. (2020). Legal protection against forest areas to ensure habitat wildlife in the Wallacea region. *IOP Conference Series: Earth and Environmental Science*, *473*(1). https://doi.org/10.1088/1755-1315/473/1/012060

Mayasari, A., Christita, M., & Suryawan, A. (2018). Keragaman Jamur Makroskopis di Arboretum Jenis-Jenis Pohon Asal Wallacea BP2LHK Manado. *WASIAN*, *5*(2), 105–114.

Muchtar, M., & Wellem, K. A. (2025). A Web Based 8-Puzzle Educational Game for Recognizing Sulawesi's Endemic Animals Using the BFS Algorithm. *Jurnal Informatika Dan Teknik Elektro Terapan*, *13*(3S1), 1773–1781. https://doi.org/10.23960/jitet.v13i3S1.8092

Nasruddin, Ismail, & Adnan. (2023). Analisis Awal Kebutuhan Pengembangan Buku Keanekaragaman Hayati Spesies Endemik Wallacea Yang Terdapat di Taman Nasional Rawa Aopa Watumohai (Tnraw), Sulawesi Tenggara. *Prosiding Seminar Nasional Biologi: Inovasi Sains & Pembelajarannya*, *23*, 110–117.

Pranoto, E. A., Susetyorini, R. E., & Prihanta, W. (2015). Identifikasi Burung di Kepulauan Kai Maluku Tenggara. *Prosiding Seminar Nasional Pendidikan Biologi 2015 Program Pendidikan Biologi Fakultas Keguruan Ilmu Pendidikan Universitas Muhammadiyah Malang*, 762–773.

Rifky, S., Kharisma, L. P. I., Afendi, H. A. R., Napitupulu, S., Ulina, M., Lestari, W. S., Maysanjaya, I. M. D., Kelvin, K., Sinaga, F. M., Muchtar, M., & others. (2024). *Artificial Intelligence: Teori dan Penerapan AI di Berbagai Bidang*. PT. Sonpedia Publishing Indonesia.

Rustan, F. R., Syaiful, M., Karim, R., Muchtar, M., Sari, J. Y., Phradiansah, P., Jamaluddin, I. I., Bantun, S., Muchlis, N. F., Pasrun, Y. P., & Sagala, L. O. H. S. (2022). Training on making creative learning media using the canva application for MI and MTs Al-Mu'minin Kendari Teachers. *Community Empowerment*, *7*(8), 1338–1343. https://doi.org/10.31603/ce.7106

Safrizal, S., & Rozak, A. (2019). PERANCANGAN GAME PUZZLE 8 BERBASIS ANDROID MENGGUNAKAN ALGORITMA A STAR. *Jurnal Ilmiah Fakultas Teknik LIMIT'S*, *15*(1), 17–23.

Sari, J. Y., Pasrun, Y. P., Muchtar, M., Karim, R., & Saputra, R. A. (2024). Development of Multimedia-Based Online Learning Media for Class VI Students of SDN 105 Kendari. *MEKONGGA: Jurnal Pengabdian Masyarakat*, *1*(1), 1–6. https://doi.org/10.69616/mekongga.v1i1.170

Struebig, M. J., Aninta, S. G., Beger, M., Bani, A., Barus, H., Brace, S., Davies, Z. G., De Brauwer, M., Diele, K., Djakiman, C., Djamaluddin, R., Drinkwater, R., Dumbrell, A., Evans, D., Fusi, M., Herrera-Alsina, L., Iskandar, D. T., Jompa, J., Juliandi, B., … Supriatna, J. (2022). Safeguarding Imperiled Biodiversity and Evolutionary Processes in the Wallacea Center of Endemism. In *BioScience* (Vol. 72, Number 11, pp. 1118–1130). Oxford University Press. https://doi.org/10.1093/biosci/biac085

Wicaksono, I. S., & Trisnawan, P. H. (2021). *Implementasi Multipath Routing menggunakan Algoritme Iterative Deepening Depth First Search pada OpenFlow Software-Defined Networking*. *5*(2), 572–581. http://j-ptiik.ub.ac.id

Wijayanti, R., Nugraha, W., & Kusrini, K. (2020). Optimalisasi Penyelesaian Permainan pada Game Puzzle 8 dengan Perbandingan Algoritma A* dan Greedy. *Citec Journal*, (1), 10–19.

Wulandari, A., Sari, R. Y., & Sulistyaningsih, D. (2023). Perbedaan Mamalia di Sulawesi dan di Sumatera dari Sudut Pandang Biodiversitas. *Jurnal Pengelolaan Sumberdaya Alam Lingkungan Wilayah Pesisir*, *1*(1), 1–8. https://journal.bengkuluinstitute.com/index.php/JEMMIES