Analysis of A Algorithm Optimization and Breadth First Search in the Water Teapot Game*

Bonifacius Vicky Indriyono[1], Widyatmoko[2]

[1,2]*Information System, Dian Nuswantoro University, Central Java, Indonesia*

[1]bonifaciusvicky@gmail.com (*)

[2]atmoko.widy06@gmail.com

Received: 2022-05-27; Accepted: 2022-07-18; Published: 2022-07-27

Abstract— Artificial Intelligence (AI) is a branch of computer science that focuses on learning how computers can work or perform as well as humans or even better than humans. The development of artificial intelligence can be used as a benchmark for technological developments. Many problems can be solved using this artificial intelligence concept. One of them is a water jug logic game. The water jugs problem is a game requiring converting a situation into another desired situation using a certain set of operations. The objective of this game is for players to develop their deductive reasoning skills by figuring out how to move x litters of water from one teapot to another. To accomplish this, they will refer to the beginning value that has to be filled in each teapot and the end value that needs to be filled in each teapot. Using the A* algorithm and the Breadth-First Search (BFS) algorithm, which both fall under the umbrella of artificial intelligence, it is possible to find a solution to this issue. The A* algorithm is a Best First Search algorithm that combines Uniform Cost Search and Greedy Best-First Search, while the BFS algorithm is a graph search algorithm that performs a wide search at each level. This study analyzed the effectiveness of the A* and BFS algorithms in solving the water jug logic problem, both from the number of steps and the time required for completion. The results of the tests were carried out, and it is known that the BFS algorithm is more effective than the A* algorithm because the BFS algorithm can provide a number of alternative solutions to the water jug logic problem with various conditions.

Keywords— Artificial Intelligence, Water Teapot Game, A* Algorithm Optimization, Breadth First Search Algorithm.

I. INTRODUCTION

Artificial Intelligence (AI) is a branch of computer science that focuses on learning how computers can work or perform as well as humans or even better. Artificial intelligence is a concept that can formulate and make a conclusion to system users based on the problems encountered [1]. According to [2], artificial intelligence is part of the technology used to solve complex problems using more humane machines. These human ways tend to lead to system activities that duplicate human behavior and thinking and apply them as algorithms that computers understand. At the same time, according to [3], Artificial Intelligence is a branch of computer technology to simulate human intelligence into a computer device. To be able to solve problems as humans do. The advantages of the artificial intelligence concept are that it is permanent, easy to use, repair, modification, and more consistent and faster in doing tasks [4]. According to [5], the scope of artificial intelligence includes natural language processing, robotics, expert systems, game applications, and navigation systems. An example of a problem that can be solved using the principles of artificial intelligence is the water jugs problem.

The logic of the water jug is one example of a common problem that has existed for a long time and sometimes still exists in human life today. The water jug logic problem is one of the most discussed examples of issues in the concept of applying artificial intelligence. In principle, this water jug problem requires conversion from one state to another expected state using a number of operating processes [6]. The water jug problem aims to fill a container of water of known volume with water that is entirely full using the help of two or more other containers of known volume but not measuring. In practice, this problem may have no solution or even the possibility of having more than one solution to solve it. Sometimes there are many ways to solve these problems, but of the many available methods, choosing the most optimal method will require its technique. The water jug logic requires users to hone their logical thinking skills in solving them using the optimal and fastest steps. The logic of this water jug can be described in the following example. For example, there are two water jugs where each teapot has a volume of n1 and n2 liters of water. The logic that must be solved from this problem is how the steps are carried out to obtain a volume of x liters of water by utilizing the available teapots. In simple terms, the steps that can be taken to solve the problem are filling a water jug, and a water jug is emptied and inserting the contents of one water jug into another water jug. In theory, it is quite short, but when faced directly with this problem, many system users are constrained by the logic of the steps that must be taken, especially if the water jug used is more than two teapots with various volumes of water.

Several previous studies discussed the problem of the logic of this water jug. It analyzes the BFS algorithm in the concept of artificial intelligence in solving water teapot problems. It explains how the process of searching and finding solutions by applying the BFS technique is described [1]. The application of the A* algorithm to optimize the search for dynamic solutions for water teapots [6]. A written journal describes how the search for the A* algorithm is to find solutions to the water jug problem [6]. The research [7] uses the BFS algorithm to find the shortest solution to solve the water jug problem. The BFS algorithm technique is used in

DOI: http://dx.doi.org/ 10.25139/inform.v7i2.4484

searching and tracing nodes and finding the fastest solution for solving water teapot logic problems [8]. The paper [9] explained solving the water jug problem using an arithmetic algorithm approach. A Cognitive Approach to Solving Water Jugs Problem describes various cognitive approaches to solving water jugs problems [10]. The result of research [11] describes the technique of solving a water jug using a non-heuristic algorithm approach to the General Two Water Jugs Problem,

Based on the previous research literature and the existing problem, namely how to determine the optimal and fastest step to solve the water teapot problem, this study will discuss the analysis of settlement techniques using the A* and BFS algorithm in detail and clearly.

II. Research Methodology

According to [12], the research method is an effort made by someone systematically following the methodological principles, such as direct observation systematic, controlled and based on existing theory and reinforced by existing symptoms. In contrast, aaccording to [13], the research method can be formulated as a scientific method used by researchers to obtain data for various purposes and specific purposes.

A. *Research methods*

The research method used in this study is a qualitative approach with a descriptive method. In this method, the data collection method is carried out in 3 stages. The first stage is to analyze how the water jug logic game works and how to solve each problem. The second stage is to test the running of this water jug logic game by using 2 teapots, 3 teapots, 4 teapots or 5 water teapots. The test uses two comparison algorithms, the BFS and A* algorithms. The third stage is to conduct a literature study to examine and study the results of previous research by researchers and a literature study in the form of books that are still related. The steps taken in this research method are shown in Figure 1.

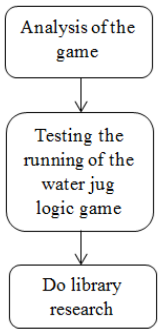


Figure 1. The Research Methods Steps

Analysis of the game: In this first stage, observations were made on how the actual water jug logic game worked and how the process of solving each problem was faced. The analysis is carried out in-depth and records relevant things found in the game as a basis for designing applications.

Testing the running of the water jug logic game: the second stage was conducted to test the running of this water jug logic game using two algorithms, BFS and A*. The purpose of the test is to see the effectiveness of the two algorithms used so that they can be used as a reference for a suitable algorithm to solve the water jug logic game. Effectiveness includes the tracing process, the number of completion steps, and the completion time.

Do library research: the third stage is a step taken to complete the knowledge and basics related to the water jug logic game, the BFS, and A* methods. The process is carried out by reading and reviewing the thoughts of previous researchers as outlined in journals, proceedings, and books related to the research topic. The study's results will later be implemented into research results in the form of a water jug logic game application product.

B. *Collecting Data Method*

Data collection techniques are a method used by researchers to obtain data to support their research. This technique refers to data sources that will be obtained either through primary sources in the form of interviews and observations and secondary sources in the form of literature studies (books, journals, and proceedings) [14]. In research activities, data collection is the main thing because one of the objectives of conducting research is to obtain data.

From the understanding of the data collection techniques above, in this study, the data were obtained from secondary sources through books, journals, and proceedings that discussed the logic of the water jug, the BFS algorithm technique, and the A* algorithm. The data obtained will be used as a reference in analyzing the logic of the water teapot with the BFS and A* algorithms, including as testing material for the manufacture of research products in the form of a water teapot logic game application.

C. *Breadth First Search Algorithm*

According to [15], there are two kinds of traversal algorithms in graphs, namely the BFS and the Depth First Search (DFS) algorithms. Although these two algorithms have the same goal, namely to visit every vertex in the graph, the approaches taken by both are different, so the application of each algorithm is also other according to the needs of the problem.

Conceptually, the BFS algorithm is a technique in the process of tracing and searching a graph. There are two main processes in the BFS algorithm: visiting a vertex in the graph and examining the vertex [26].

The BFS algorithm in the process requires a queue (*queue = q)* which is used to accommodate the nodes that have been searched. Some of these traced nodes are needed to serve as guidelines for tracing several other nodes adjacent to the first node. Each node that has been traced is stored once in the queue. In addition to requiring a queue of q, BFS also requires a boolean Table that is used to store the nodes that have been visited so that no node is visited more than once [15].

The BFS is a graph search algorithm [16]. The BFS is considered an algorithm that becomes a guideline for other

DOI: http://dx.doi.org/ 10.25139/inform.v7i2.4484

graph search algorithms. The working principle of the BFS algorithm is to perform a search on the threshold between the nodes that have been traced and the nodes that have not been traced. BFS will trace the vertices so that every vertex that has a distance from the initial vertex will be traced before tracing a vertex that has a distance of j+1 from the initial vertex. The search process carried out by BFS is to search each node from the beginning. The BFS uses a queue structure to run the expansion process [17]. According to the research [15], the BFS algorithm has the following working process:

- Enter the starting node into the queue (queue).
- Take the first node from the queue, then check whether it is an answer completion or not.
- If the answer is a settlement, then the search process ends, and the results will be returned.
- If it is considered a non-completion answer, then all nodes close to that node are stored in a queue.

According to [18], the BFS algorithm is a search algorithm that starts with the root node (Level 0) to go to the next level. The search process is carried out by searching all nodes with the same level until the final result is determined. Otherwise, it will move to the next level. The flow chart of the BFS technique is shown in Figure 2.

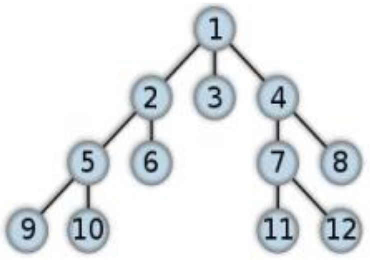


Figure 2. Breadth First Search algorithm flowchart

Based on Figure 2, how find the fastest route can be described briefly as:

- The initial position leads to the node on the right, the 2nd node.
- Then, the process is continued by the node adjacent to the 2nd node, namely the 3rd node, and then heading back to the other adjacent node, namely the 4th node.
- The search position leads to a node adjacent to the 4th node, but because there are no adjacent nodes, the current position returns to node no.2 and goes to the next node, namely node 5.
- The search is carried out repeatedly until it leads to node 8

So from the explanation above, by applying the BFS method, the fastest route to pass is 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8.

D. Definition of A Algorithm*

A* is a search algorithm that functions to perform input analysis, evaluate a number that could be a solution, and produce a solution. According to [19], A* is an algorithm widely used in the graph transversal process and pathfinding. So that the route can be found from the initial node to the destination node, a search is carried out using a graph.

The A* algorithm is an improvement from the Best First Search (BFS) algorithm by modifying its heuristic function, which is widely used for searching and searching for the shortest route. Similar to the function of the BFS algorithm in finding solutions, the way A* works is also assisted by a heuristic function where this function plays a role in regulating the order in which nodes will receive the first visit. The heuristic function is a function that will guide A* to find a solution by assigning a value to each node [20].

The A* algorithm is defined as the Best First Search algorithm where this algorithm is a combination of the Uniform Cost Search algorithm with the Greedy-Best First Search [21]. In practice, the Uniform Cost Search algorithm functions to determine the smallest distance from the initial node to the next node to the destination node, while the Greedy-Best First Search algorithm is an algorithm that utilizes a Heuristic function that is used to estimate the cost from the initial node to the destination node. This heuristic function plays an important role in controlling the search in the A* algorithm so that in the end, through this algorithm, a complete path will be found, and a solution will always be found if there is an optimal solution [22].

The A* algorithm is the most widely used algorithm in finding paths and transverse graphs (nodes) [23]. This algorithm can be written in the following mathematical Equation (1).

$$f(n)=g(n)+h(n) \tag{1}$$

where *f(n)* variable is *n* node/point evaluation function. The *g(n)* variable is the distance of the coordinate point to the destination point, and the *h(n)* variable is the heuristic value between coordinate points A* algorithm contains multiple lists, namely open list and closed list. According to [24], there are three conditions for each node being explored: the state when it is entered in the open list, the state when it is entered in the closed list, and the state when it is outside in the open list and closed list. Based on Equation (1), the value of g(n) is the coordinate distance to the destination point, where the value for g(n) is obtained by multiplying the distance on the map with the map scale. This can be written in the following Equation (2) [25].

$$g(n) = \textit{distance on map x map scale} \tag{2}$$

To determine the heuristic value/*h (n)* can be written with the following Equation (2).

$$h(n) = \sqrt{\left(x_n - x_{goal}\right)^2 + \left(y_n - y_{goal}\right)^2} \tag{3}$$

Where the *h(n)* variable is the value heuristic for point *n*, the *Xn* variable is the value of x coordinate of point *n*, the *Yn* variable is the value of *y* coordinate of point *n*, and the *Xgoal* variable is the x-coordinate value of the destination point. The *Ygoal* variable is the value of the y-coordinate of the destination point.

DOI: http://dx.doi.org/ 10.25139/inform.v7i2.4484

The working principle of this A* algorithm is to run a transversal on each node one by one to get the closest path. A* algorithm will calculate the distance of one of the paths, and then the path is saved, and then calculates the distance of the other path. If the calculation for all paths has been completed, then the algorithm A* algorithm closest path will be chosen by A* algorithm [21]. The work process flow diagram of the A* algorithm is shown in Figure 3.

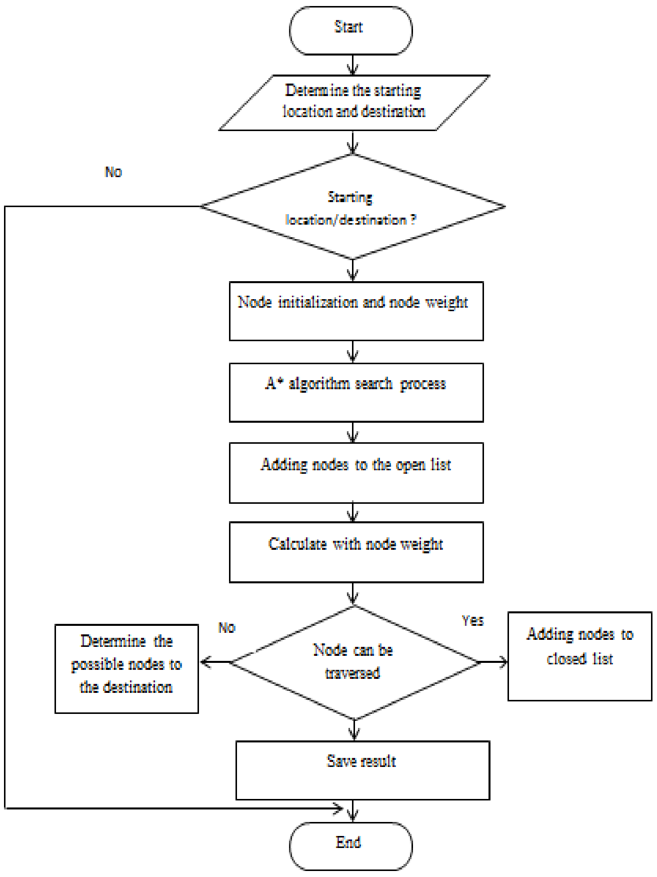

Start
Determine the starting location and destination
Starting location/destination ?
No
Node initialization and node weight
A* algorithm search process
Adding nodes to the open list
Calculate with node weight
Node can be traversed
No
Yes
Determine the possible nodes to the destination
Adding nodes to closed list
Save result
End

Figure 3. Algorithm process flowchart A*

1) Determine the starting location and destination: First, determine the initial position of the node and at which position the node must stop (destination location).

2) Starting location/destination: After determining the initial location and destination, then the next step is to check whether the initial location and destination are appropriate? If yes then.

3) Node initialization and node weight: At this stage, node initialization and node weight determination are carried out. This is done to make it easier to trace the path of the traversed nodes and determine which nodes are included in the open list or close list

4) A algorithm search process:* Initialization and node weighting, the tracing process is carried out with the concept of A*.

5) Adding nodes to the open list: In the search results that have been carried out, it can be found that the nodes that have been visited and the heuristic values have been calculated but have not been selected as the best nodes. This open list contains nodes that still have a chance to be chosen as the best node.

6) Calculate with node weight: After entering the visited nodes into the open list, the heuristic value will be calculated based on the previously initialized weights.

7) The node can be traversed: The results of the calculation of the value that has been done, the question will arise whether the node can be passed?

8) Determine the possible nodes to the destination: If the node cannot be passed, it will be re-determined the node that is still possible to get to the destination

9) Adding nodes to the closed list: If the nodes can be traversed, they are included in the closed list and considered as nodes that have been tested or traversed.

10) Save result: The last step is to save the results, which is to store a list of nodes that are the shortest alternative paths.

III. Result and Discussion

In this section, we will test the running of the water jug logic game using the Breadth First Search and A* algorithms. The test was carried out using two water jugs.

A. *Analysis Techniques With A* Algorithm*

The following example is given to solve the water jug logic problem with the concept of the A* algorithm. There are 2 water teapots labeled T1 (Teapot 1) and T2 (Teapot 2), where each teapot has a water capacity of 8 and 6 liters, respectively, and there is one water faucet. The initial conditions of all water jugs are empty. The problem that must be solved is: how to make one of the two water jugs contain 4 liters of water, while the T1 teapot has a size of 8 liters of water and the T2 teapot has a 6 liters capacity.

1) Define the scope of the problem: From the case example above, it can be defined as a problem that can be symbolized by the symbol (v1, v2), where v1 is the volume of water filled in the T1 teapot as much as 8 liters so that it is written v1 = 0.1,2,3,4, 5,6,7 or 8, while v2 is the volume of water filled in the T2 teapot so that it can be written v2 = 0.1,2,3,4,5, or 6.

2) Determine the initial state value and the final destination value: The initial value state is that the contents of both the teapots T1 and T2 are empty (0,0). The final value is how to fill one teapot with exactly 4 liters of water regardless of the volume of water filled in the other so that the final goal value can be represented by (x,4) for each x value.

3) Establish production rules: The determination of this production process can be illustrated by drawing a tree structure model for all the conditions described above. By assuming the initial state value (0,0) as the initial node of a

DOI: http://dx.doi.org/ 10.25139/inform.v7i2.4484

tree structure, it will be possible to trace the next few nodes that can still occur. This production rule means a process capable of changing from one state to another. The process of tracing this node continues until all the rules that occur are obtained. An answer will not be found if it does not completely define all production processes. The detailed determination of the production process that can be applied to solving the water jug problem is shown on Table I.

TABLE I
PRODUCTION PROCESS TABLE

No	Production Rules	Information
1	*(v1,v2), v1 < 8*	*(8,v2).* Fill the water jug T1
2	*(v1,v2), v2 < 6*	*(x,6).* Fill the T2 water jug.
3	*(v1,v2), v1 > 0*	*(0,v2).* The contents of the T1 water jug are drained so empty.
4	*(v1,v2), v2 > 0*	*(v1,0).* The contents of the T2 water jug are drained so empty.
5	*(v1,v2), v1 + v2 ≥8 && v2 > 0*	*(8, v2-(8-v1)).* The contents of the water in the T2 teapot are poured into the full T1 teapot.
6	*(v1,v2), v1 + v2 ≥6 && v1 > 0*	*(v1-(6-v2),v2).* The water in the T1 teapot is poured into the T2 teapot until it's full.
7	*(v1,v2), v1 + v2 ≤8 && v2 > 0*	*(v1+v2,0).* All of the water that is in the T2 teapot is poured into the T1 teapot
8	*(v1,v2), v1 + v2 ≤6 && v1 > 0*	*(0,v1+v2).* All of the water in the T1 teapot is poured into the T2 teapot.

After the general production rules are defined, the next step will be to find the optimal solution using the A* algorithm. The solution with the A8 algorithm can be described using the binary tree structure of the A* algorithm. The sequence of the tracing process with a binary tree to get a possible solution to a situation is shown in Figure 4.

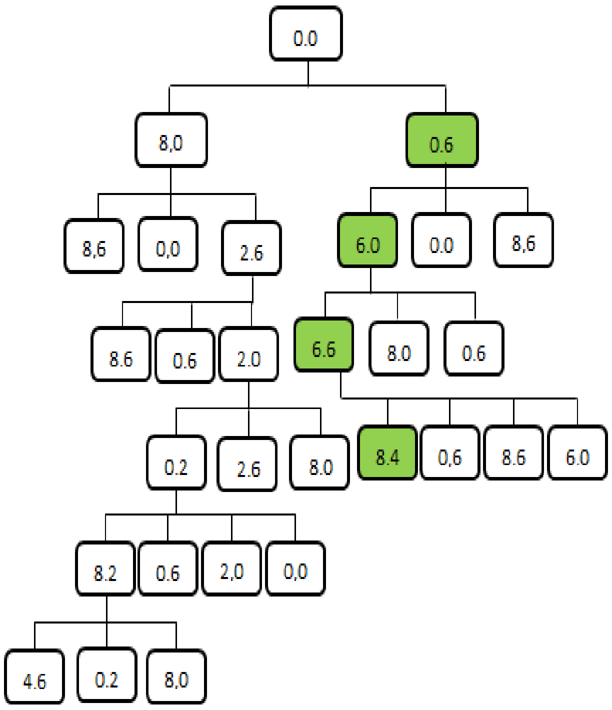


Figure 4. A* Algorithm Binary Tree Search

From the search using the binary tree structure of the A* algorithm above, it can be determined 4 optimal steps for the solution, namely: (0-0),(0-6),(6-0),(6-6),(8-4). The following Table II shows the results of the 4 optimal steps.

TABLE II.
OPTIMAL STEP SOLUTION TABLE

	Amount of Water Teapot T1	Amount of Water Teapot T2	(V1, V2)	Production Rule Number
0	0	0	0-0	0
1	0	6	0-6	2
2	6	0	6-0	7
3	6	6	6-6	2
4	8	4	8-4	5

B. *Analysis Techniques with Breadth First Search (BFS) Algorithm*

The BFS has a search concept; namely, all nodes for level l will be visited first before tracing the next nodes located at level l+1. The search starts from left to right from the lowest node and continues to the first level (level 1), then will move to the next level. This process will continue to be repeated until a solution is found. An overview of the BFS search can be seen in Figure 5.

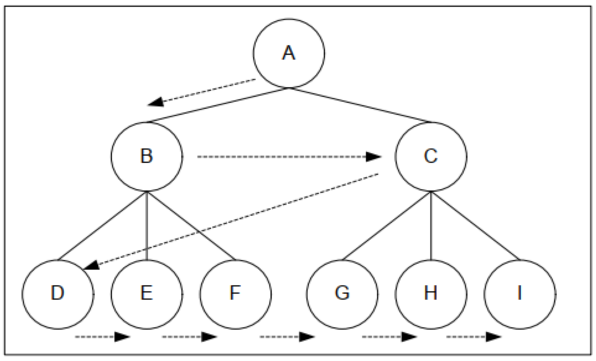


Figure 5. BFS Algorithm Search Concept

In Figure 5, it can be seen that the search process starts from the initial condition, node A. Departing from node A, two more nodes are branched, which will be used as child nodes, namely node B and node C. Then, the tracing process is continued to node B. B is branched by three new nodes, namely nodes D, E, and F, and a search to node C will result in nodes G, H, and I. This search process will continue until a solution is found. Because the flow of the BFS algorithm search process is to observe each node at each level before the movement that leads to a deeper level, then initially, all of these conditions will be achieved through the shortest path from the initial state. Because of this, it is possible to guarantee that this search process will find the shortest path from the initial state to the destination state.

The following illustration can be used to demonstrate how the BFS algorithm can be applied to the process of analyzing the solution to the water-teapot logic problem. There are 2 water teapots labeled T1 (Teapot 1) and T2 (Teapot 2), where each teapot has a full capacity of 8 and 6 liters of water and one water faucet. The initial conditions of all water jugs are empty. The problem must be solved: how to fill 4 liters of water into one of the available water jugs, while the T1 teapot has a full 8-liter capacity and the T2 teapot has a full 6-liter water capacity. The solution to this problem can be found using the tree structure of the BFS algorithm, as shown in Figure 6.

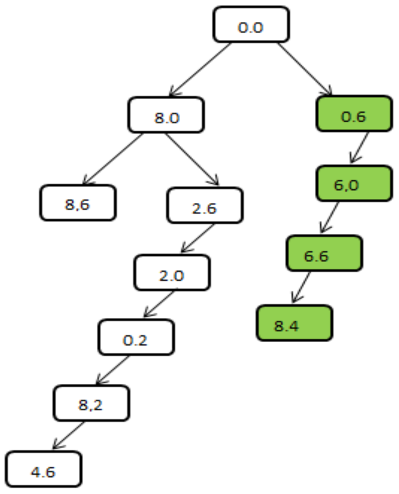


Figure 6. BFS Tree Solution Search

Based on Figure 6, to solve the water teapot logic case above, two solutions can be found, namely: the first solution using 4 steps and the second solution using 8 completion steps. The following is a description of each solution step solution.

1. The solution with 4 steps can be described as follows:
   - Fill teapot 2 (T2) with water to the brim so that the current state becomes (0.6).
   - Pour water from teapot 2 (T2) to teapot 1 (T1) so that the state becomes (6.0).
   - Fill the teapot 2 (T2) with water until it is full so that the situation becomes (6.6).
   - Pour water from teapot 2 (T2) into teapot 1 (T1) so that the situation becomes (8,4).
2. The solution with 6 steps can be described as follows:
   - Fill the T1 teapot with water until it is full so that the state becomes (8.0).
   - Pour water from teapot T1 into teapot T2, so the situation becomes (2,6).
   - Empty the contents of the T2 teapot. The current state becomes (2.0).
   - Pour water from teapot T1 into teapot T2. The current state becomes (0.2).
   - Fill the T1 teapot to the brim. The current state is (8,2).
   - Pour water from teapot T1 into teapot T2. (4,6).

From the description above, it can be seen that to solve the case of how to make one of the two water jugs filled with 4 liters of water using the BFS algorithm, we found two optimal solution steps. And it can also be seen that the first solution, which has 4 optimal steps, results in the same as the optimal step traced using the A* algorithm.

C. System Testing

System testing is intended to prove that the A* algorithm and the BFS algorithm can be implemented properly to find the optimal step solution in solving water teapot logic problems with different conditions. The following is an example of a water teapot logic case that will be implemented into the system and tested for correctness.

There are 2 teapots with labels T1 and T2, where each teapot has sizes N1 and N2 liters of water. Of the two teapots, one of them will be filled with X liters of water. The teapots have a capacity of 4 liters for T1 and 3 liters of water for T2. The problem is how to fully fill 2 liters of water into teapot A, whose actual water capacity is 4 liters. To solve the above case, we first define the general steps as follows:

- Perform early identification of problems: The problem is symbolized using *(x1,x2)*. The *X1* variable is the content of water in teapot T1, and T2 is the content of water in teapot T2.
- Initial and final destination conditions: The initial conditions are teapots T1 and T2 are all empty (0,0), and The final destination is a teapot filled with water with a capacity of 2 liters: *(2,x)* for any value of <u>*x*</u>.
- Regulation: The rules for solving this water teapot problem in Table III.

TABLE III
SETTLEMENT RULE TABLE

No	Initial conditions	Destination condition
1	*(x1,x2), x1 < 4*	*(4,x2)*. Fill the teapot T1.
2	*(x1,x2), x2< 3*	*(x1,3)*. Fill the T2 teapot.
3	*(x1,x2), x1 > 0*	*(0,x2)*. The contents of the T1 teapot are emptied by removing all the water.
4	*(x1,x2), x2 > 0*	*(x1,0)*. The contents of the T2 teapot are emptied by removing all the water.
5	*(x1,x2), x1 + x2 ≥ 4 && x2 > 0*	*(4, x2-(4-x1))*. Water from the T2 teapot is poured into the T1 teapot until it is full.
6	*(x1,x2), x1 + x2 ≥ 3 && x1 > 0*	*(x1-(3-x2),x2)*. Water from the T1 teapot is poured into the T2 teapot until it's full.
7	*(x1,x2), x1 + x2 ≤ 4 && x2 > 0*	*(x1+x2,0)*. All the water from the T2 teapot is poured into the T1 teapot.
8	*(x1,x2), x1 + x2 ≤ 3 && x1 > 0*	*(0,x1+x2)*. The contents of the T1 teapot are poured completely into the T2 teapot

Based on the Table of production rules above, it can be described tracing with the help of a tree as shown in Figure 7.

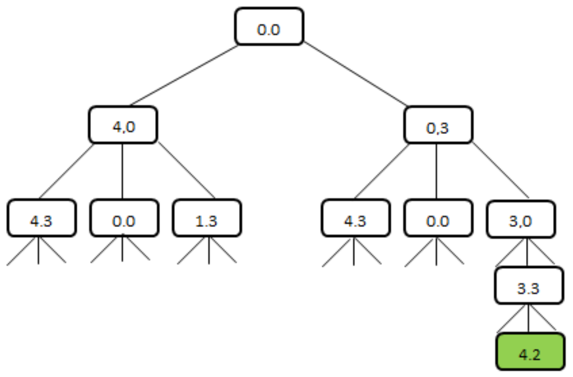


Figure 7. Solution search tree

Based on Figure 7, it can be concluded that a 4-step solution is shown in Table IV.

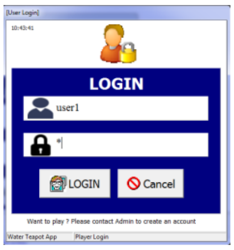
TABLE IV
SOLUTION TRACING TABLE

Fill Water Teapot A	Fill Water Teapot B	Rule Number	Information
0	0	2	The initial condition of teapot A and teapot B
0	3	7	Fill teapot B with 3 liters of water
3	0	2	Pour the contents of teapot B into teapot A
3	3	5	Fill teapot B with 3 liters of water
4	2		Pour the contents of teapot B into teapot A

The system test results from the examples given show that both the A* algorithm and the BFS algorithm can actually be used to search, and both work optimally in finding the right solution for each problem in the water jug logic.

D. *Water Teapot App Simulation*

After analyzing and testing the running of the A* and BFS algorithms in solving water teapot logic problems, the next step is to simulate the water teapot logic problem solving using an application. There are several pages available in this water teapot logic application. Player login page: This form records the players' names who will use this water teapot application. If the player has not received an account, the player can register an account through this login page. The login page is shown in Figure 8.



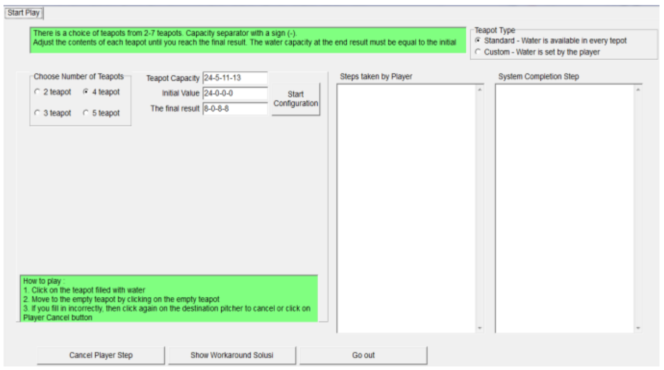
user1
*

Want to play ? Please contact Admin to create an account

Water Teapot App | Player Login

Figure 8. Login page

After the player gets an account, then the player can enter a user name and log in. And if it's true, the start page of the water teapot game will be displayed, as shown in Figure 9.



There is a choice of teapots from 2-7 teapots. Capacity separator with a sign (-).
Adjust the contents of each teapot until you reach the final result. The water capacity at the end result must be equal to the initial

Teapot Type: [✓ Standard - Water is available in every tepot] Custom - Water is set by the player

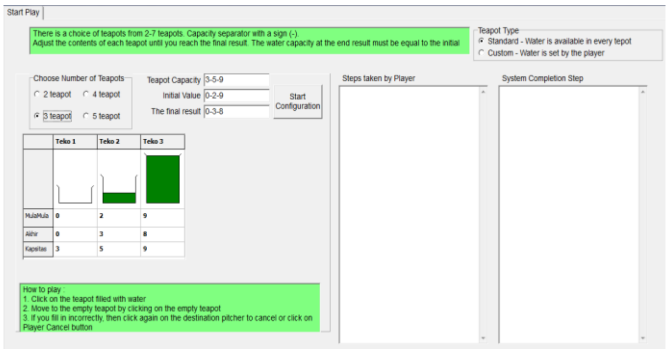
Choose Number of Teapots: 2 teapot [✓ 4 teapot] 3 teapot 5 teapot

Teapot Capacity: 24-5-11-13
Initial Value: 24-0-0-0
The final result: 8-0-8-8

How to play :
1. Click on the teapot filled with water
2. Move to the empty teapot by clicking on the empty teapot
3. If you fill in incorrectly, then click again on the destination pitcher to cancel or click on Player Cancel button

Figure 9. Login Page

Game Page: On this page, players are asked to determine the number of teapots, the initial value of the volume of water, and the final volume of the teapot's contents. After everything is filled in, the player can press the start configuration button. The results of the settings on this page are shown in Figure 10.



There is a choice of teapots from 2-7 teapots. Capacity separator with a sign (-).
Adjust the contents of each teapot until you reach the final result. The water capacity at the end result must be equal to the initial

Teapot Type: [✓ Standard - Water is available in every tepot] Custom - Water is set by the player

Choose Number of Teapots: 2 teapot 4 teapot [✓ 3 teapot] 5 teapot

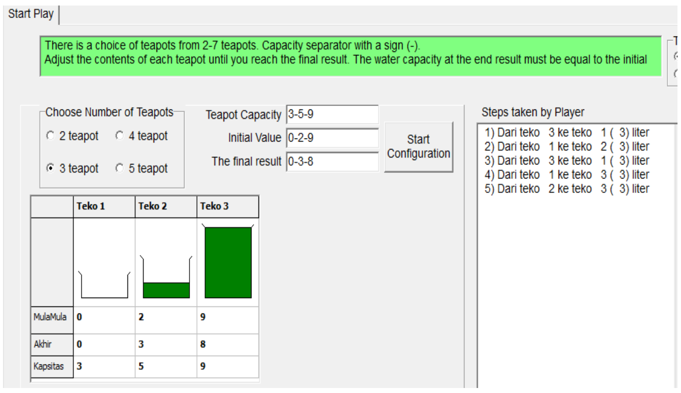
Teapot Capacity: 3-5-9
Initial Value: 0-2-9
The final result: 0-3-8

	Teko 1	Teko 2	Teko 3
MulaMula	0	2	9
Akhir	0	3	8
Kapsitas	3	5	9

How to play :
1. Click on the teapot filled with water
2. Move to the empty teapot by clicking on the empty teapot
3. If you fill in incorrectly, then click again on the destination pitcher to cancel or click on Player Cancel button

Figure 10. Game Page

Player step identity: The identity of the player's steps in solving the problem will be displayed in the "Steps taken by player" window.



There is a choice of teapots from 2-7 teapots. Capacity separator with a sign (-).
Adjust the contents of each teapot until you reach the final result. The water capacity at the end result must be equal to the initial

Choose Number of Teapots: 2 teapot 4 teapot [✓ 3 teapot] 5 teapot

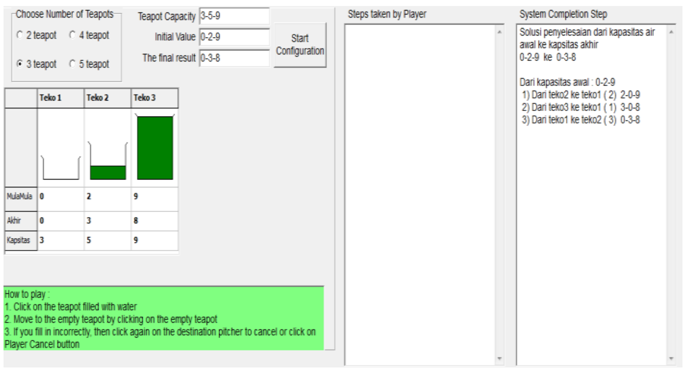
Teapot Capacity: 3-5-9
Initial Value: 0-2-9
The final result: 0-3-8

	Teko 1	Teko 2	Teko 3
MulaMula	0	2	9
Akhir	0	3	8
Kapsitas	3	5	9

Steps taken by Player:
```
1) Dari teko  3 ke teko  1 (  3) liter
2) Dari teko  1 ke teko  2 (  3) liter
3) Dari teko  3 ke teko  1 (  3) liter
4) Dari teko  1 ke teko  3 (  3) liter
5) Dari teko  2 ke teko  3 (  3) liter
```

Figure 11. The Identity Of The Player's Steps

Solution steps: In this section, the system will display the steps for solving the water teapot problem being worked on. This step is based on the running of the algorithm used in the research



Choose Number of Teapots: 2 teapot 4 teapot [✓ 3 teapot] 5 teapot

Teapot Capacity: 3-5-9
Initial Value: 0-2-9
The final result: 0-3-8

	Teko 1	Teko 2	Teko 3
MulaMula	0	2	9
Akhir	0	3	8
Kapsitas	3	5	9

How to play :
1. Click on the teapot filled with water
2. Move to the empty teapot by clicking on the empty teapot
3. If you fill in incorrectly, then click again on the destination pitcher to cancel or click on Player Cancel button

System Completion Step:
```
Solusi penyelesaian dari kapasitas air
awal ke kapsitas akhir
0-2-9  ke  0-3-8

Dari kapasitas awal : 0-2-9
 1) Dari teko2 ke teko1 ( 2)  2-0-9
 2) Dari teko3 ke teko1 ( 1)  3-0-8
 3) Dari teko1 ke teko2 ( 3)  0-3-8
```

Figure 12. System Steps Solution

DOI: http://dx.doi.org/ 10.25139/inform.v7i2.4484

IV. CONCLUSION

Based on the results of comparative analysis and testing of the water teapot logic case using the A* and BFS algorithm. Algorithm A* and BFS can optimally provide a solution for solving water teapot logic problems. The number of steps generated from both the A* and BFS algorithms is the same. The only difference is that the BFS algorithm can provide an alternative to the number of other steps compared to the A* algorithm. The BFS method can provide multiple alternative numbers of completion steps, which can be utilized as an option to solve this water jug logic problem. As a result, the BFS algorithm can be used to solve this problem.

The suggestions that can be given to improve the process and the results of this research for the next process it is necessary to do analysis and testing not only using the A* or BFS algorithm but can also use other search algorithms so that the results can be much more optimal. The results of the implementation of this study are still limited to the case of 5 water jugs. So, for optimal results, adding cases with more teapots is necessary.

V. FUTURE WORK

Applications resulting from this research are still limited to solving conditions with 5 water jugs. So, for further development, more water jugs will be added so that the accuracy of the application of this search algorithm will be much more leverage

REFERENCE

[1] Wijaya,E., "Analysis of the Use of the Breadth First Search Algorithm in the Concept of Artificial Intelligence", *TIME Journal*, Vol. II No 2, p. 18-26, 2013, ISSN : 2337 – 3601

[2] Dedi, N, Sri, W. "Development of Learning Media for Tracking Systems in Artificial Intelligence Subjects Based on Mutimedia". *Journal of Informatics Engineering*. 2014; Vol. 2 No. 1; p. 738-748.

[3] Muclisin, R. *Artificial intelligence*. Yogyakarta: Graha Ilmu. 2019.

[4] Suyanto. *Artificial Intelligence: Search, Reasoning, Planning and Learning*. Bandung : Informatika. 2014.

[5] Widodo, B. AI For Beginners. Yogyakarta : Andi Offset. 2014.

[6] Harianja, F., "Application of the A* Algorithm to Optimization Problems for Dynamic Water Solution Search". *Pelita Informatika Journal*, Vol. IV, No. 3, August, 2013, ISSN : 2301-9425

[7] Siagian, E.R., "Water Jug Simulation Software Using the Breadth First Search Method". *Pelita Informatika Journal*, Vol. 6, No. 2, Oct, 2017, ISSN 2301-9425, p. 206-209.

[8] Parlaungan, T.F.,; Tugimin, I.," Software for Searching for Solutions to Water Bottle Problems (Waterjug Problem) Using the Breadth First Search (BFS) Algorithm". *Journal of Information and Communication Technology*. Vol. 2 No. 1, Oct, 2021, ISSN: 2252-4517.

[9] Man, Y.K., "Solving the General Two Water Jugs Problem via an Algorithmic Approach". *Proceedings of the International MultiConference of Engineers and Computer Scientists.* IMECS. Vol I, March., 2015.

[10] Saxena, D., et al., "A Cognitive Approach to solve Water Jugs Problem". *International Journal of Computer Applications.* Vol124, No.17, August ,2015 .

[11] Man, Y.K., "A Non-heuristic Approach to the General Two Water Jugs Problem". *Journal of Communication and Computer 10*. July, 2013, p. 904-908.

[12] Sukardi., *Competency Education Research Methodology and Practice*. Jakarta: PT Bumi Aksara. 2011.

[13] Sugiyono. *Quantitative, Qualitative and R&D Research Methods*. Bandung : Alfabeta. 2013.

[14] Suharsimi, A. *Research Procedure A Practice Approach Revised Edition*. Jakarta : Rineka Cipta. 2010.

[15] Zai, et al, "Shortest Route Simulation of Tourism Locations in Nias Using Breadth First Search and Tabu Search Methods.", *InFact Journal*, Vol. 2 No. 1. 2016

[16] Wahyuni, E., et al., "Three-dimensional Explosion Crack Modeling For Serious Games", *National Seminar on Information Technology and Multimedia*. STMIK AMIKOM Yogyakarta.. ISSN : 2302-3805. P. 13-1-13.6. 2013

[17] Alkindi, U., et al.," Implementation of the Breadth First Search Algorithm in Pacman to Regulate Character Movement.", *Journal of Computer Research (JURIKOM),* Vol. 5 No. 6. 2018

[18] Sutojo, et al, *Artificial Intelligence*. Yogyakarta: Andi. 2011.

[19] Setiawan, K., et al,." Calculating the Shortest Route Using A* Algorithm with Euclidean Distance Function.", *National Seminar on Information and Communication Technology* (SENTIKA). Yogyakarta. 2018

[20] Nugraeni, R.A.., et al," Application of the A* Algorithm in Completing the Shortest Route for the Distribution of Goods.", *Unnes Journal of Mathematics*. Vol. 4 No. 1. p-ISSN 2252-6943 e-ISSN 2460-5859.

[21] Hermanto, D.; Dermawan, S., "Application of the A-Star Algorithm as the Shortest Route Finder on the Hexapod Robot". *National Journal of Electrical Engineering,* Vol. 7, No. 2, p-ISSN: 2302-2949, e-ISSN: 2407 – 7267. 2018

[22] Taufiq, P.J., et al.," Implementation and Analysis of the A*(Star) Algorithm to Determine Paths With Multiple Goals in Npc(Non-Playable Character) Movements". *e-Proceeding of Engineering*. Vol.2, No.,3. p. 7800. ISSN : 2355-9365. 2015

[23] Nuryoso, Y.H., et al., "Application of the A* Algorithm in Searching for the Shortest Route on Angkot Routes in Sukabumi City". *Journal of Bachelor of Informatics Engineering*. Vol. 8, No. 1., e-ISSN 2338-5197, p. 21-35. 2020

[24] Suyanto. *Artificial Intelligence Search, Reasoning, Planning and Learning*. Bandung : Informatics.2014

[25] Pramudhita, A.C,; Muljono. "City Nearest BRT Stop Search System Application Semarang Using Android-Based A* Method". *Resti Journal (System Engineering and Information Technology),* Vol. 2 No.1, p.430-436. 2018.

[26] Indriyono, B.V, ;Widyatmoko, “Optimization of Breadth-First Search Algorithm forPath Solutions inMazyin Games”. *International Journal of Artificial Intelligence & Robotics (IJAIR).* Vol 3, No. 2, e-ISSN 2686-6269, p. 58-66. 2021

DOI: http://dx.doi.org/ 10.25139/inform.v7i2.4484