Constructing a Decision Tree to Find the Best Starting Hole in Indonesian Traditional Game 'Congklak': Maximizing the Number of Seed in Gudang by the End of the Turn

Nicholas Reymond Sihite - 13522144[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13522144@std.stei.itb.ac.id

***Abstract*— *Congklak* is an Indonesian traditional game played by two players where the goal is to maximize the number of seeds each player has in *gudang* by the end of the game. Strategy is very important to win this game, therefore the starting hole is crucial. Determining the best starting move to maximize seeds count in *gudang* can be done by constructing a decision tree for each hole options. One way to efficiently construct a decision tree for this problem is simulating the game into a C program using a type of data structure called linked list with circular buffer. Without limiting the amount of time a player can pick another hole after landing the last seed in *gudang*, it is possible to get 73 seeds or even higher (limit is currently unknown). However, if players are limited to three repetitions, the maximum number of seed is 28.**

***Keywords*— *congklak*, decision tree, *gudang*, maximizing, starting hole.**

I. Introduction

Indonesia is a country well known for its diverse backgrounds, cultures, and heritages. One of Indonesia's intangible cultural heritage is a traditional game called *Congklak* (*congkak*, *dhakon*). This game is played by two players on a board with 14 small holes and 2 large holes (*gudang*, *rumah*) where each player gets 7 small holes and 1 large hole.

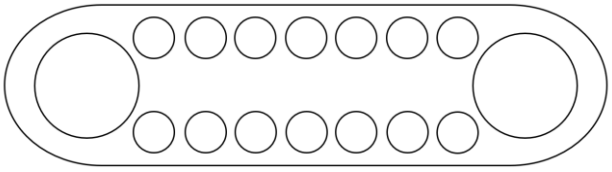

Fig. 1 Congklak Board
(source: writer's archive)

Each small hole is filled with 7 seeds/stones and the goal is to take as many seeds as possible to each player's *gudang*. By the end of the game, the player with the most number of seed in his/her *gudang* wins the game. Similar to other competitive game, it requires strategy to win a *congklak* match. A lot of *congklak* players have their 'lucky hole' to start with, which they believe would bring them to victory. In truth, do these 'lucky hole's really exist? To answer that question, the meaning of 'lucky hole' should first be defined. Since the goal of this game is to take as many seeds as possible to *gudang*, a 'lucky hole' can be defined as a hole the player start with, that results in the most possible number of seeds in *Gudang* by the end of that player's turn. This paper answers the "which hole is the luckiest hole to start with?" question by constructing a decision tree for each hole taken.

II. Theoretical Basis

A. What is a Tree?

In the context of informatics, a tree is a connected-undirected-graph which does not have any circuits [1], [2]. There are terms for objects in a tree such as root, leaf, subtree, depth, etc. A root is a vertex without any predecessor vertices. A leaf is a vertex without any successor vertices. A subtree is a tree that can be seen as a fraction of the main tree. Tree's depth is a number that represents how 'deep' the tree is.

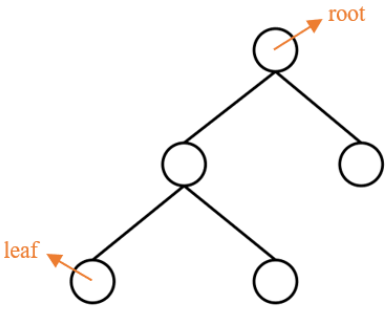

Fig. 2 Tree with a depth of 2
(source: writer's archive)

B. Decision Tree

There are a lot of different applications of trees. One of them is called a decision tree. Exactly like its name, decision trees are used to determine actions based on the decisions. The leaves of this type of tree usually represent what action should be taken, hence the name decision tree.

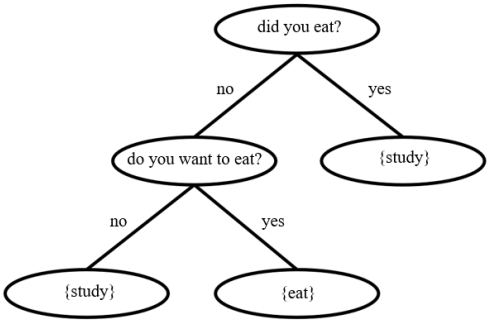

Fig. 3 Decision Tree to Determine What Action Should be Taken
(source: writer's archive)

C. Rules of Congklak

Congklak is a traditional game that has been in society for a very long time. Different areas have different conventions on specific cases that might be encountered in the game. Putting aside any possible differences, the following rules are the ones used in this paper [3].

1. The board is divided into two areas.
2. Each player gets seven small holes in the area closest to them and one *gudang* on the left side of the board in each player's perspective.
3. Seven seeds are put in each small holes.
4. Players determine which one should go first (method used is left up to the player).
5. For each turn, the current player can pick seeds from one of said players' non-empty small hole(s).
6. All seeds taken should be distributed one by one in a clockwise order for every hole (small and *gudang*), except for the opposite player's *gudang*.
7. There are a few conditions about what a player should do upon finishing seed(s) distribution:
   a. If the distribution ends in any non-empty small hole, the player can take all seeds in that hole then continue the distribution,
   b. If the distribution ends in own player's empty small hole (before putting the final seed), the player can take the only seed in his hole and all seeds in the exact opposite hole then put all of it in the player's own *gudang*,
   c. If the distribution ends in opponent's empty small hole (before putting the final seed), the player's turn ends, and
   d. If the distribution ends in own player's *gudang*, the player can choose another small hole from his side to start another distribution. This condition only applies to the **first three repetitions**. If in the fourth distribution the player still ends up in *gudang*, said player's turn will end.
8. After a player, A, finishes their turn, the other player, B, takes their turn, then A, then B, and so on until there are no more seeds in small holes in their area.
9. The player with the most number of seeds in *gudang* by the end, wins the game.

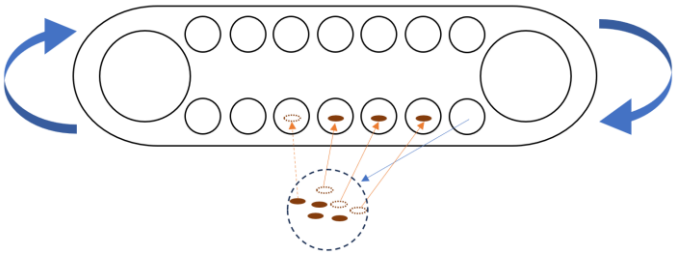

Fig. 4 Congklak Game Simulation Starting from Hole 1
(source: writer's archive)

III. Constructing the Decision Tree

A. Problem Representation in the Decision Tree

Before starting the decision tree construction, defining the representation must be done first. In this decision tree, every vertex represents how many seeds there are in *gudang* and every edge represents which hole was taken.

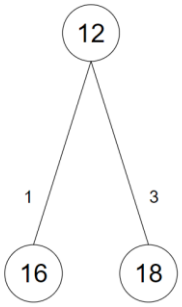

Fig. 5 Congklak Decision Tree Example 1
(source: writer's archive)

As an example, in *Fig. 5* the initial number of seeds in *Gudang* is show in the tree's root, which is 12. The player can choose to take seeds from hole 1 or 3 to start in this turn. By choosing hole 1, the number of seeds in *gudang* increases to 16 and by choosing hole 3, it increases to 18. Notice how there are no options to take hole 2, 4, 5, 6, and 7. That means that there are no seeds in those holes, therefore they do not appear as the edges.

To handle cases mentioned in rule 7d, if the player chooses a path that fulfils that specific condition, the vertex that the edge leads to is highlighted as red.

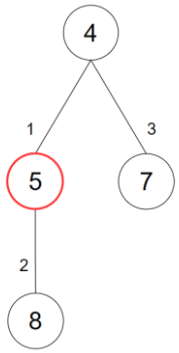

Fig. 6 Congklak Decision Tree Example 2
(source: writer's archive)

In *Fig.* 6, there is one vertex colored red which is a result of taking seeds from hole 1. That means by taking seeds in hole 1, the player ended his seed distribution in *gudang*. According to rule 7d, for the first three repetitions of that condition, the

current player will get to choose another hole to restart the distribution. In this example, the player can only take seed(s) from hole 2 to continue the distribution that ends with 8 seeds in *gudang*. Analyzing this example, it can be concluded that while initially the number of seeds in *gudang*, if hole 1 is taken, is less than if hole 3 is taken, because rule 7d, the final number of seeds in *gudang* by the end of the turn would be maximized if hole 1 is taken.

B. Congklak First Turn Simulation in C

Calculating the number of seeds in *gudang* for each hole taken would be an inconvenience if it is done manually. To check all possible holes means resetting the seed count in the board for every hole. While still possible, constructing the decision tree in this manner would take a long time.

One other step that can be (and was) taken to avoid that inconvenience is creating a program that simulates the first turn of *congklak*. Notice how in a *congklak* match, players must distribute the seeds one by one from one hole to the next hole (linked) in a clockwise order (circular). This nature is similar to that of a data structure called linked list with circular buffer, therefore this data structure can be (and is) used in the program. The program takes two inputs: one is how many seeds the player wants in each small hole initially; and two is which hole would the player want to start with. The following is the realization of the program in C programming language as a void function.

```
void congklakFirstTurn(){
    /* LOCAL VARIABLES */
    List Congklak;
    Address currHole, gudang;
    int seedsPerHole, startingHole, holeContent,
currSeeds, i, j;
    boolean endofTurn;

    /* ALGORITHM */
    printf("How many seeds do you want to put in
each hole?\n");
    printf(">> "); scanf("%d", &seedsPerHole);
    createCongklak(&Congklak, seedsPerHole);
    printf("Here is a Congklak board with %d seeds
in each hole:\n", seedsPerHole);
    displayCongklak(Congklak);
    printf("Which hole do you want to start
with?\n");
    printf(">> "); scanf("%d", &startingHole);

    // find starting hole
    i = 1;
    currHole = FIRST(Congklak);
    while (i < startingHole){
        currHole = NEXT(currHole);
        i++;
    }
    holeContent = INFO(currHole);
    printf("Hole %d currently has %d seeds\n",
startingHole, holeContent);
    printf("Simulating Congklak game first turn
from hole %d...\n\n", startingHole);

    // simulate congklak first turn
    currSeeds = holeContent;
    INFO(currHole) = 0;
    endofTurn = false;
    currHole = NEXT(currHole);
    gudang = gudangAddress(Congklak);
    i++;

    while (!endofTurn){
        if (i != 16){ // if not in enemy's gudang
            // move one seed to current hole
            currSeeds--;
            INFO(currHole)++;

            if ((currSeeds == 0) && (currHole ==
gudang)){ // ended up in own gudang, repick hole
(not endofTurn)
                displayCongklak(Congklak);
                displayCurrentTurn(i);
                printf("you ended up in gudang,
you can repick a hole to start again\n");
                printf(">> "); scanf("%d",
&startingHole);
                j = 1;
                currHole = FIRST(Congklak);
                while (j < startingHole){
                    currHole = NEXT(currHole);
                    j++;
                }
                holeContent = INFO(currHole);
                printf("Hole %d currently has %d
seeds\n", startingHole, holeContent);
                printf("Continuing Congklak game
first turn from hole %d...\n", startingHole);
                currSeeds = holeContent;
                INFO(currHole) = 0;
                i = j;
            } else if ((currSeeds == 0) &&
(INFO(currHole) == 1) && (i < 8)){ // ended up in
an own empty hole, take own seeds and opposite's
seeds (endofTurn)
                displayCongklak(Congklak);
                displayCurrentTurn(i);
                displaySeeds(currSeeds);
                printf("you ended up in an empty
hole in your territory, you take your own seeds
and opposite's seeds\n");
                printf("you take %d seed(s) from
your hole number %d\n", INFO(currHole), i % 8);
                currSeeds = INFO(currHole);
                INFO(currHole) = 0;
                j = i;
                while (j != 16 - i){
                    currHole = NEXT(currHole);
                    j++;
                }
                printf("you take %d seed(s) from
enemy hole number %d\n", INFO(currHole), j % 8);
                currSeeds += INFO(currHole);
                displaySeeds(currSeeds);
                INFO(currHole) = 0;
                INFO(gudang) += currSeeds;
                printf("You moved %d seed(s) to
your gudang\n", currSeeds);
                currSeeds = 0;
                endofTurn = true;
            } else if ((currSeeds == 0) &&
(INFO(currHole) == 1) && (i > 8)){ // ended up in
an enemy empty hole (endofTurn)
                printf("you ended up in an empty
hole in enemy's territory, you stop\n");
                endofTurn = true;
            } else if ((currSeeds == 0) &&
(INFO(currHole) > 1)){ // ended up in a not empty
own hole (not endofTurn)
                currSeeds = INFO(currHole);
                INFO(currHole) = 0;
            }

            // display congklak board and turn
            printf("Current congklak board:\n");
            if (i > 8){
                displayCurrentTurn(i);
            }
            displayCongklak(Congklak);
            if (i <= 8){
                displayCurrentTurn(i);
            }
```

```
            displaySeeds(currSeeds);
        }

        if (!endofTurn){
            if (i == 16){
                printf("Skipping enemy's
gudang...");
            }
            printf("\n\n----------------------
moving on to the next hole ----------------------
\n\n");
            currHole = NEXT(currHole);
            i++;
            if (i > 16){
                i = 1;
            }
        }
    }
    printf("Congklak game first turn simulation is
finished\n");
    printf("Here is the Congklak board after the
first turn:\n");
    displayCongklak(Congklak);
    printf("The number of seeds in your gudang is
%d!\n\n", INFO(gudang));
    printf("\n\n---------------------- end of first
turn ----------------------\n\n");
}
```

Fig. 7 Implementation of Congklak First Turn Simulation in C
(source: writer's archive)

Inside the congklakFirstTurn function, there are five additional functions used, which are createCongklak, displayCongklak, displaySeeds, displayCurrentTurn, and gudangAddress. Implementation of those five functions are not shown in this paper.

C. Program in Action

The simulation is implemented as a void function. To run the program, user would need to create a simple driver shown in *Fig. 8*.

```
int main(){
    congklakFirstTurn();
}
```

Fig. 8 Driver for congklakFirstTurn
(source: writer's archive)

After compiling and starts running, the program would ask for the number of seeds to be put in each hole (*Fig. 9*).

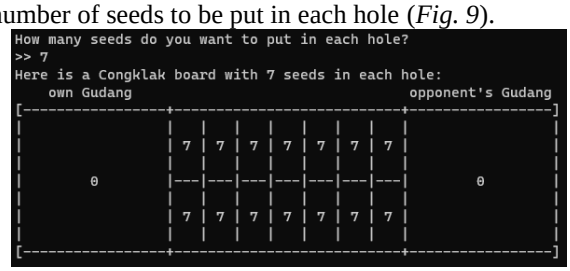

Fig. 9 congklakFirstTurn Program in Action (1)
(source: writer's archive)

Next (*Fig. 10*), the program would ask for which hole does the player want to start with (for instance, input 1). After that, the program would show the simulation for every distribution one by one until there are no more seeds.

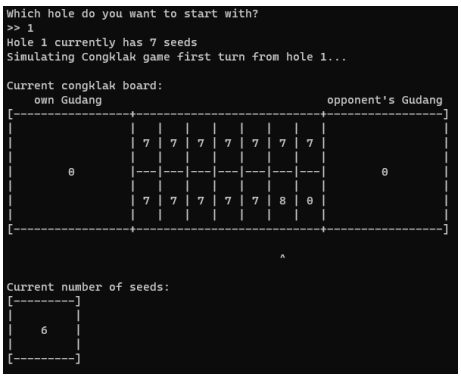

Fig. 10 congklakFirstTurn Program in Action (2)
(source: writer's archive)

Picking hole 1 at the start of the game would make the player end up in *gudang*. As a result (*Fig. 11*), the program would ask for another hole to start with (for instance, put 4).

Fig. 11 congklakFirstTurn Program in Action (3)
(source: writer's archive)

Taking seeds in hole 4 would not end the distribution in the current player's *gudang*, therefore ending the first turn.

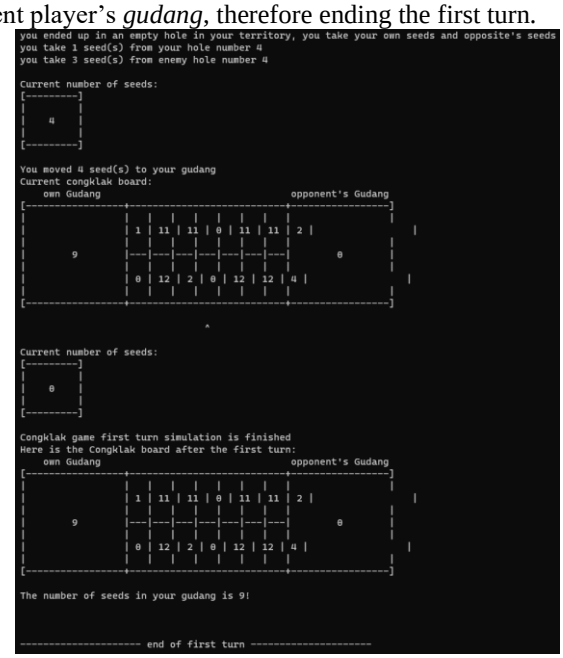

Fig. 12 congklakFirstTurn Program in Action (4)
(source: writer's archive)

Shown in *Fig.* 12, the final number of seeds in *gudang* at the end of the first turn is 9. Therefore, taking 1-4 path would result in ending the turn with 9 seeds in *gudang*.

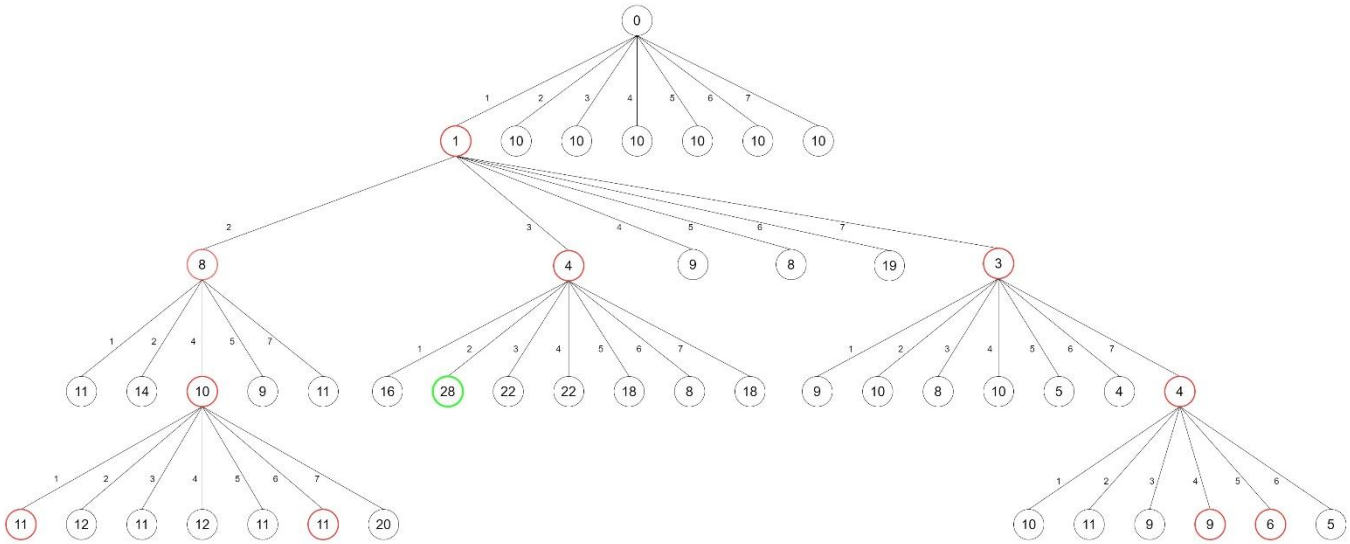

Fig. 13 Decision Tree to Find Number of Seeds in Gudang for Every Possible Move in The First Turn
(source: writer's archive)

IV. RESULT

Using the program that has been created, the decision tree to find the best/luckiest starting hole is created. *Fig. 13* shows the decision tree. The leaf highlighted with green shows the maximum number of seeds a player can get by the end of the first turn, which is 28 seeds. The route taken to get this result is 1-3-2.

However, the 'first three repetitions' in rule 7d does not actually apply in real life. If a player theoretically always lands in *gudang*, the player can repick another hole for as many times as the player wants, not limited to 3. The additional rule was added because without limitation, the decision tree would be massive.

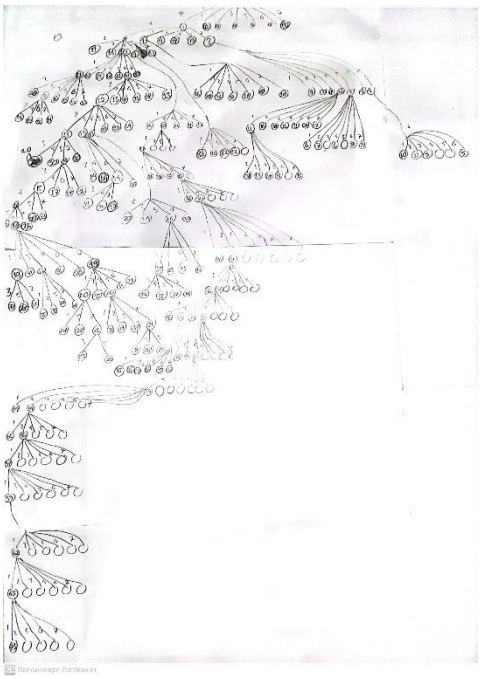

Fig. 14 Unfinished Rough Sketch of Decision Tree for Not Limited Repetitions
(source: writer's archive)

Using this unlimited method, the player can even get up to 73 seeds in *gudang* by the end of the first turn using the following route: 1-2-5-2-1-5-2-2-3-4-3-1-2-3-1-2-1-1-1.

Fig. 15 Usage of Current Known Maximum Number of Seeds in Gudang by The End of The First Turn
(source: writer's archive)

Theoretically, a player would only need 50 seeds to win $\left(\frac{98}{2}+1\right)$. The shortest found route to achieve victory condition is: 1-2-5-2-1-5-2-2-3-4-3-1-1 which results in 54 seeds in *gudang*.

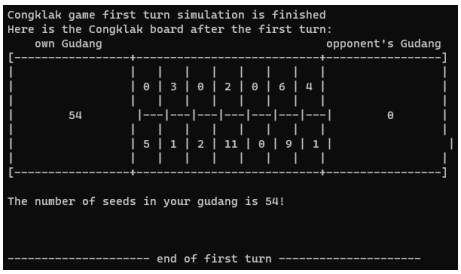

Fig. 16 Usage of Current Known Minimum Route to Win in the First Turn
(source: writer's archive)

V. CONCLUSION

The maximum number of seeds in *gudang* by the end of the first player's turn that can be achieved with 'first three repetitions' rule is 28 and the route to get it is 1-3-2. Other results can be seen in *Fig. 13*. Without limitations, it is currently

unknown what is the maximum number of seeds a player can get only in the first turn, but it is possible to get 73 seeds in *gudang* in only one turn.

VI. Acknowledgment

The writer would like to firstly thank Almighty God for His guidance in helping the writer finish this paper. Second, the writer would like to thank himself for gathering up the courage to write a paper with an interesting-nonmainstream topic and finishing it in time. Third, the writer would like to thank all lecturers of IF2120 Discrete Mathematics, especially Dr. Rinaldi Munir and Monterico Adrian, S.T., M.T. as lecturers of K03 Jatinangor for guiding the writer in discovering the beauty of the discrete world. Finally, the writer would like to thank all other (possibly unknown) parties indirectly involved in encouraging the writer to finish this paper.

References

[1] Munir, R. (2023). *Pohon (Bag. 1)*. Retrieved December 10, 2023, from https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/22-Pohon-Bag1-2023.pdf

[2] Munir, R. (2023). *Pohon (Bag. 2)*. Retrieved December 10, 2023, from https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/23-Pohon-Bag2-2023.pdf

[3] Kids, S. (2022, August). *Bermain Congklak | Permainan Tradisional Anak Indonesia | Video Belajar Anak | Video Edukasi*. [Video]. Retrieved December 10, 2023, from https://www.youtube.com/watch?v=JJAxKqjJHcQ

[4] Indonesia. Ministry of Education, Culture, Research, and Technology. (2012). *Congklak*. Retrieved December 11, 2023, from https://warisanbudaya.kemdikbud.go.id/?newdetail&detailCatat=2196

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Jatinangor, 11 Desember 2023

Nicholas Reymond Sihite 13522144