Songklanakarin J. Sci. Technol.
44 (5), 1164–1171, Sep. – Oct. 2022

http://www.sjst.psu.ac.th

Original Article

The winning percentage in congkak using a randomised strategy

Chuei Yee Chen[1], Shahrina Ismail[2], Fong Peng Lim[1], and Kai Siong Yow[1, 3*]

[1] *Department of Mathematics and Statistics, Faculty of Science, Universiti Putra Malaysia, UPM Serdang, Selangor, 43400 Malaysia*

[2] *Faculty of Science and Technology, Universiti Sains Islam Malaysia, Bandar Baru Nilai, Negeri Sembilan, 71800 Malaysia*

[3] *School of Computer Science and Engineering, College of Engineering, Nanyang Technological University, 639798 Singapore*

Received: 6 August 2021; Revised: 26 April 2022; Accepted: 6 July 2022

Abstract

Congkak is a traditional counting game played in Southeast Asia including Malaysia, Singapore, Brunei and Indonesia. To start a game, a board that has 16 holes together with 98 marbles are required. Each player controls a set of seven holes and own a store. The winner of the game is the player who captured more marbles into the store at the end of the game. Note that the first-move advantage exists in chess; we investigate if the first-move advantage holds in congkak also. We model the route for each player in congkak using a directed graph and adopt these graph representations in our programs, to compute the winning percentage of each player. We focus on games between novices, hence a randomised strategy is used in our algorithm. We present the first experimental results for 100,000 games between novices in congkak. We also suggest some questions for future research in this area.

Keywords: mancala, congkak, counting game, first-move advantage, randomised algorithm, winning percentage

1. Introduction

Congkak is a Southeast Asian mancala game that was likely introduced to Southeast Asia by Indian or Arab traders in the 15th century. This traditional counting game is named *congkak* in Malaysia and Brunei, *congka* or *congklak* in Indonesia, and *sungkâ* in the Philippines. Some have claimed that the term congkak originated from an old Malay term *congak* that conveys mental arithmetic as the meaning. In congkak, a player who is able to predict the moves through mathematical calculations is believed to have an advantage to win the game.

Congkak is played by two players on a wooden boat-shaped board that has 16 cup-shaped holes—two sets of seven holes (Some congkak boards have two sets of five holes or two sets of nine holes. However, the one with two sets of seven holes is the most common.), together with two larger holes known as the *stores* at each end (Figure 1).

Figure 1. A congkak board with two sets of seven holes

Each player controls the seven holes on their side, and possesses a store, in a congkak game. There are 98 marbles (or cowrie shells or tamarind seeds) used in a congkak game.

*Corresponding author
Email address: ksyow@upm.edu.my

At the beginning of each game, seven marbles are placed in each hole except in the two stores of the congkak board. The winner of the game is the one who has captured more marbles into the store at the end of the game. The detailed rules of the game can be found in Section 2.

Note that there is an ancient African board game, namely *awari*, that is very similar to congkak. The game has drawn much attention from researchers in computer science and even artificial intelligence, and it was solved by using parallel retrograde analysis (Romein, 2003). Romein and Bal (2002) have also shown that awari ends in a draw with a perfect play, by performing an extensive search using a parallel computer with 144 processors.

The rules and moves in congkak are not as complicated as in other board games including Go and chess (Farr, 2017; Fraenkel & Lichtenstein, 1981; Newell, Shaw, & Simon, 1958; Storer, 1983). A digital congkak simulation was developed using a combination of neural networks and Min-Max algorithm (Chepa, Alwi, Din, & Safwan, 2013), and the win ratio of up to 100 games using these methods was investigated. Note that there is only limited literature on congkak compared to some other board games. Hence, people may wonder if there is any advantage for one particular player in a congkak game.

In chess, there is a consensus that the player who makes the first move has an inherent advantage. The overall *winning percentage* in chess is calculated by summing up the percentage of games won plus half the percentage of games drawn. Statistical results show that *White* (the player who moves first in chess) has a greater winning percentage, ranging between 52% and 56%, at most levels of play (Wikipedia, 2020). This advantage is however less significant in blitz games and games between novices. Some of the results for chess are shown in Table 1.

In this article, we determine the winning percentage for games between novices in congkak by using a program, given that statistical results in congkak are relatively limited. The winning percentage obtained is then used as an indicator for comparison purposes. Two novices are assumed to contest in a random way (no counting involved), hence a randomised strategy is used in the simulation program. We examine if the first-move advantage holds in congkak, for games between novices, by providing some experimental results.

2. Rules of Congkak

The game is set up by first placing seven marbles in each small hole (except for the two stores) in a congkak board. The initial position of the congkak board is illustrated in Figure 2.

Note that there are different versions of rules for playing congkak (Tan, 2020; Vtaide, 2020). We use the following rules, with Figure 3 illustrating the moves visually.

1. A player ***A*** is assigned randomly as the first player, whereas the other player is the second player denoted by ***B***. The player ***A*** owns the red holes and the player ***B*** owns the blue holes. The two larger holes at the end of both sides represent the stores.
2. To begin the game, ***A*** chooses one of the seven red holes on his side, and empties all the marbles in the hole. The marbles collected are distributed in an anticlockwise order, where one marble is dropped into other holes except the store of ***B***. The distribution process is called *sowing*. For example, in Figure 3, the red directed cycle and the blue directed cycle represent the routes of ***A*** and ***B***, respectively. It is clear that each route has 15 holes.
3. The location of the final marble during the sowing process determines how the game continues. There are four possible scenarios as follows (assume that ***A*** is the current player):
   - If the final marble falls into the player's own store, the player chooses one (non-empty) hole on his side and the sowing continues.
   - If the final marble falls into a non-empty hole, the hole is emptied and the sowing continues.
   - If the final marble falls into an empty (red) hole on the player's side, the player collects all the marbles from the opponent's (blue) hole that is opposite to the empty hole. All the collected marbles together with the final marble are put into the player's store. The player forfeits his turn and it is the opponent's turn to continue the game. If the opposite hole is empty, the player forfeits his turn and it is the opponent's turn to continue the game.

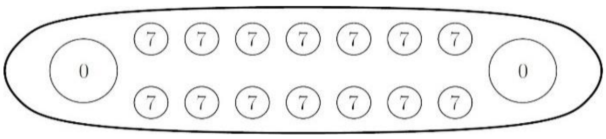

Figure 2. The initial position of a congkak game, where each number represents the number of marbles in each position

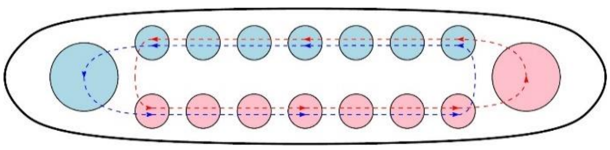

Figure 3. The routes of players ***A*** and ***B***, highlighted in red and blue, respectively.

Table 1. Some statistical results for chess (Chessgames, 2020 & Streeter, 1946)

Events	White wins	Drawn	Black wins	Winning percentage of White
Tournaments (1851-1932)	38.12%	30.56%	31.31%	53.40%
Chessgames.com (1475-2020)	37.79%	34.08%	28.13%	54.83%

- If the final marble falls into an empty (blue) hole on the opponent's side, the player forfeits his turn and it is the opponent's turn to continue the game.

4. If there is no marble left on the player's side when it is his turn, then he must pass.
5. The game ends when all the marbles are moved into the stores. The player who captures more marbles is the winner. It is a tie game if the players captured the same number of marbles.

The game can be continued in the second round by redistributing marbles from players' own stores to their own holes. Each player starts to fill their own holes starting from the rightmost hole (the one that is closest to the store). If the player has more than 49 marbles in the store, put the extra marbles back to the store. If the player has less than 49 marbles, some of the holes will either be empty or have less than seven marbles. All these holes are *burnt* and are left empty. The leftover marbles will be returned to the store.

Once the board is set up, the game is continued by bypassing the burnt holes, based on the previous rules (Note that some impose one additional rule in the second round onwards, where the marble that accidentally falls into a burnt hole will be confiscated and put into the opponent's store.). The game repeats until one player loses all his holes or concedes defeat.

3. Graph Representations

As shown in Figure 3, graphs can be used to represent congkak. We provide such a representation in this section.

A *directed graph* (or *digraph*) G consists of a non-empty finite set $V(G)$ of elements called *vertices*, and a finite family $A(G)$ of ordered pairs of elements of $V(G)$ called *arcs*. A *vertex-weighted digraph* D is a digraph whose vertices v are associated with real weights, that is $w : V(D) \to \mathbb{R}, v \mapsto w(v)$. We call $w(v)$ the *weight* of the vertex v. All weights are non-negative in any vertex-weighted digraph.

The route for each player in congkak can be represented by using a vertex-weighted directed cycle that has 15 vertices. Two such vertex weighted directed graphs are highlighted in blue and red in Figure 3.

Let D be a vertex-weighted directed graph. Suppose $V(D) = \{A_i, B_i\}$ for $i \in \{1,2,...,8\}$. Clearly, the number of vertices $|v(D)| = 16$ and $v(D)$ corresponds to the holes and stores of a congkak board, as shown in Figure 4. The directed cycles of players ***A*** and ***B*** are represented as follows:

$C_A = (A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8, B_1, B_2, B_3, B_4, B_5, B_6, B_7)$,
$C_B = (B_1, B_2, B_3, B_4, B_5, B_6, B_7, B_8, A_1, A_2, A_3, A_4, A_5, A_6, A_7)$.

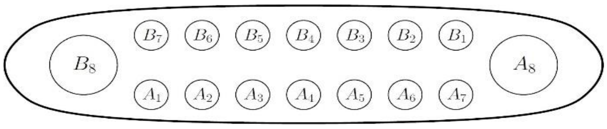

Figure 4. A labelled congkak board

If we replace the label of each vertex in C_A and C_B by its weight, the initial position for each player in a congkak game is as follows:

$C_A = (7, 7, 7, 7, 7, 7, 7, 0, 7, 7, 7, 7, 7, 7, 7)$,
$C_B = (7, 7, 7, 7, 7, 7, 7, 0, 7, 7, 7, 7, 7, 7, 7)$.

Likewise, we have

$C_A = (0, 0, 0, 0, 0, 0, 0, T_A, 0, 0, 0, 0, 0, 0, 0)$,
$C_B = (0, 0, 0, 0, 0, 0, 0, T_B, 0, 0, 0, 0, 0, 0, 0)$
as the final positions where $T_A + T_B = 98$.
We adopt these representations in our program in Section 4.

4. Experimental Results

We study if there is any advantage for one specific player in congkak, particularly the first-move advantage that exists in chess. We focus on games between novices, by using the assumption that no counting occurs throughout the game, i.e., players are not allowed to memorize the number of marbles in each hole for counting purposes.

We let the player ***A*** be the first player, and since no counting is allowed, each player picks their moves randomly (hence the randomised strategy) in the game. Note that we determine the winner of the game based on the result obtained in the first round, given that the mechanism of the game in the remaining rounds is similar. The algorithms of our programs are shown in Algorithms 1-4 (see Tables 3-6). By implementing some minor modifications, these algorithms can be used to determine the winner of the game in the remaining rounds.

Let w be the number of games won by a player out of t games in congkak. The *winning percentage* of the player is w/t. Given that the nature of tie games in congkak is quite different from chess, we exclude the percentage of tie games in our calculation.

We use Python 3 in conducting the experiments. We determine the winning percentage of each player in congkak, based on the experimental results that are derived by using 100,000 games. One sample output of a congkak game can be found in Appendix. We split the games into five categories, where each category consists of 20,000 games. The overall winning percentage is calculated by using the average percentage of all these categories. The statistical results of the 100,000 games are summarised and shown in Table 2.

Table 2. The experimental results of 100,000 games between novices in congkak

Category	***A*** wins	***B*** wins	Tie	Winning percentage of ***A***	Winning percentage of ***B***	Percentage of tie games
I	12249	6962	789	61.25	34.81	3.94
II	12122	7105	773	60.61	35.53	3.86
III	12152	7110	738	60.76	35.55	3.69
IV	12212	6990	798	61.06	34.95	3.99
V	12121	7090	789	60.60	35.45	3.95
Total	60856	35257	3887	60.85	35.26	3.89

Table 3. Algorithm 1

Algorithm 1 Congkak(*aList*)

Input: A list *aList* that represents the initial position of the directed cycle of the first player in congkak
Output: The winner of the game

```
bList ← [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
totalA, totalB, currentPlayer, currentList ← 0,0, A, aList
randomStart ← True
while totalA + totalB ≠ 98 do
    if randomStart = True then
        if the first seven elements in currentList are not all zero then
            currentHole ← RandomHole(currentList)
        else
            if currentPlayer = A then
                UpdateList(currentList, aList, bList, totalA, totalB)
                currentPlayer, currentList ← B, bList
                continue
            else
                UpdateList(currentList, bList, aList, totalB, totalA)
                currentPlayer, currentList ← A, aList
                continue
                      currentList, currentHole ← Sowing(currentList, currentHole)
    if currentHole = 7 then
        randomStart ← True
        totalA, totalB ← aList[7], bList[7]
        continue
    else if currentHole < 7 then
        if currentList[currentHole] = 1 then
            randomStart ← True
            if the hole H that is opposite to currentHole is not empty then
                Put the final marble and all the marbles in H into the current player's store
                if currentPlayer = A then
                    UpdateList(currentList, aList, bList, totalA, totalB)
                    currentPlayer, currentList ← B, bList
                    continue
                else
                    UpdateList(currentList, bList, aList, totalB, totalA)
                    currentPlayer, currentList ← A, aList
                    continue
            else
                if currentPlayer = A then
                    UpdateList(currentList, aList, bList, totalA, totalB)
                    currentPlayer, currentList ← B, bList
                    continue
                else
                    UpdateList(currentList, bList, aList, totalB, totalA)
                    currentPlayer, currentList ← A, aList
                    continue
        else
            randomStart ← False
            continue
    else
        if currentList[currentHole] = 1 then
            randomStart ← True
            if currentPlayer = A then
                UpdateList(currentList, aList, bList, totalA, totalB)
                currentPlayer, currentList ← B, bList
                continue
            else
                UpdateList(currentList, bList, aList, totalB, totalA)
                currentPlayer, currentList ← A, aList
                continue
        else
            randomStart ← False
            continue
if totalA − totalB > 0 then
    print The player A is the winner.
```

Table 3. Continued.

Algorithm 1 Congkak($aList$)

else if $totalA - totalB < 0$ **then**
    print The player ***B*** is the winner.
else
    print This is a tie game.

Table 4. Algorithm 2

Algorithm 2 RandomHole(L)

Input: A list L of integers
Output: Return an index r where $L[r] \neq 0$
$temp \leftarrow$ create an empty list
for i in L **do**
    if i is not zero **then**
        append the index of i to $temp$
Randomly pick an element r from $temp$
return r

Table 5. Algorithm 3

Algorithm 3 Sowing(L, r)

Input: A list L that contains 15 integers, and an index r
Output: Return the updated list L and the updated index r
$marbleInHand \leftarrow L[r]$
$0 \leftarrow L[r]$
$k \leftarrow r + 1$
while $marbleInHand \neq 0$ **do**
    $L[k] \leftarrow L[k] + 1$
    $marbleInHand \leftarrow marbleInHand - 1$
    $k \leftarrow (k + 1) \bmod 15$
$r \leftarrow k - 1$
return L, r

Table 6. Algorithm 4

Algorithm 4 UpdateList($tempList, oriList, modifiedList, oriTotal, modifiedTotal$)

Input: Three lists that contain 15 integers each, and two integers
Output: Return two updated lists and two updated integers
$oriList \leftarrow tempList$
$oriTotal \leftarrow tempList[7]$
$modifiedList[7] \leftarrow modifiedTotal$
Replace the first seven and the last seven elements in $modifiedList$ by the last seven and the first seven elements in $tempList$, respectively
return $oriList, modifiedList, oriTotal, modifiedTotal$

From Table 2, we can see that the first-move advantage exists for games between novices in congkak. The player ***A*** who is also the first player in these 100,000 games has a winning percentage of 60.85%. On the other hand, the winning percentage of ***B*** is just 35.26%. The rate for tie games is about 4%. If these results are compared with the statistical result for chess in Table 1, we can see that the first-move advantage is slightly more significant for games between novices in congkak.

Overall, based on the winning percentage of ***A*** in each category, we believe that the first-move advantage is approximately 60% based on the randomised strategy that is used in our algorithms. We however foresee that this advantage becomes more significant for games between experts, since players are only allowed to move following the directed cycle, and the number of vertices in the cycle is fixed. Therefore, if the first player manages to control and get the best location of the final marble in each move (according to Rule 3), the player will certainly have a higher winning rate.

5. Conclusions and Future Work

In this study, we introduced the history of congkak and the rules of this board game. We gave a graph representation of the route for each player in congkak, by using vertex-weighted directed graphs. The representation was then used in our programs to determine the winning percentage of each player by simulated runs of the game. We then presented the first experimental results for 100,000 games between novices in congkak. Under this setup, we conclude that the first-move advantage exists in the game where the winning percentage of the first player is about 60%.

Apart from the first-move advantage, people with a good memory are believed to have a higher winning percentage in congkak. They could probably count and make a better choice in order to start the sowing process. Hence, we believe that appropriate machine learning methods will always lead to a win in congkak. We now suggest some relevant questions for future research.

1. A player is an *expert* in congkak if the player is able to predict the moves through mathematical calculations. Logically, people believe that the winning percentage of an expert is higher compared to others. For a game between an expert and a novice, if the expert is also the first player, it is almost sure that the expert will win the game by taking the first-move and the counting advantage. On the other hand, if the novice starts the game, who would be the probable winner in this scenario? Does the first-move advantage or the counting advantage contribute a greater impact to the game?
2. Does the first-move advantage still hold for games between experts?

Suppose we categorize games in congkak into the following scenarios.

- *Scenario I*: games between novices.
- *Scenario II*: games between a novice and an expert where the novice is the first player.
- *Scenario III*: games between a novice and an expert where the expert is the first player.
- *Scenario IV*: games between experts.

We say that a move is *maximal* in congkak if the maximum number of marbles is deposited into the player's store when the player forfeits his turn.

3. If every move of an expert player is maximal in Scenario II, what is the lower bound of the winning percentage, given that the expert player is likely to win the game by taking the first-move and the counting advantage?
4. If every move of an expert player is maximal in Scenarios II, III and IV, what is the minimum number of moves that are required to end a congkak game?

Note that the percentage of tie games in Scenario I is about 4%. If it is a tie, players might have to restart the game in order to determine the winner.

5. Under which scenario will the percentage of getting a tie game be the highest? (We believe that this occurs in Scenario IV, when each move of each expert player is maximal.) What will be the upper bound?

Well defined graph polynomials provide a variety of information about the enumeration of graphs or digraphs (Chung & Graham, 1995; Gordon & McMahon, 1989; Tutte, 1954; Tutte, 1967; Welsh, 1999; Yow, 2019; Yow, Farr, & Morgan, 2018).

6. Does any of the existing graph polynomials relate to some graph representations of congkak? If not, does there exist one such definition?

Acknowledgements

This research was partially supported by Universiti Putra Malaysia under Fundamental Research Grant Scheme FRGS/1/2019/STG06/UPM/02/18 and Putra Grant GP-IPM/2020/9684300.

References

Chepa, N., Alwis, A., Din, A. M., & Safwan, M. (2013). The application of neural networks and min-max algorithm in digital congkak. *Proceedings of the 4th International Conference on Computing and Informatics*, 222-227.

Chessgames. (2020, April 21). Statistics page of online chess database and community. Retrieved from https://www.chessgames.com/chessstats.html

Chung, F. R. K., & Graham, R. L. (1995). On the cover polynomial of a digraph. *Journal of Combinatorial Theory, Series B, 65*, 271-290.

Farr, G. E. (2017). The probabilistic method meets Go. *Journal of the Korean Mathematical Society, 54*(4), 1121-1148. doi:10.4134/JKMS.j160360

Fraenkel, A. S., & Lichtenstein, D. (1981). Computing a perfect strategy for $n \times n$ chess requires time exponential in n. In S. Even & O. Kariv (Ed.), *Automata, Languages and Programming* (*ICALP 1981*), Lecture Notes in Computer Science, Vol. 115 (pp. 278-293). Berlin, Germany: Springer. 1981, doi:10.1007/3-540-10843-2_23

Gordon, G. P., & McMahon, E. W. (1989). A greedoid poly nomial which distinguishes rooted arborescences. *Proceedings of the American Mathematical Society 107*(2), 287-298.

Newell, A., Shaw, J. C., & Simon, H. A. (1958). Chess-playing programs and the problem of complexity. *IBM Journal of Research and Development, 2*(4), *320-335*. doi:10.1147/rd.24.0320

Romein, J. W., & Bal, H. E. (2002). Awari is solved. *ICGA Journal, 25*(3), 162-165.

Romein, J. W., & Bal, H. E. (2003). Solving awari with parallel retrograde analysis. *Computer, 36*(10), 26-33.

Storer, J. A. (1983). On the complexity of chess. *Journal of Computer and System Sciences, 27*(1), 77-100.

Streeter, W. F. (1946). Is the first move an advantage. *Chess Review*, 16-B.

Tan, B. (2020, April 21). Congkak. Retrieved from https://eresources.nlb.gov.sg/infopedia/articles/SIP_1733_2010-11-26.html

Tutte, W. T. (1954). A contribution to the theory of chromatic polynomials. *Canadian Journal of Mathematics, 6*, 80-91.

Tutte, W. T. (1967). On dichromatic polynomials. *Journal of Combinatorial Theory, 2*, 301-320.

Vtaide. (2020, April 21). Malaysia: Congkak. Retrieved from http://www.vtaide.com/ASEAN/Malaysia/congkak.html

Welsh, D. J. A. (1999). The Tutte polynomial. *Random Structures and Algorithms, 15*, 210-228.

Wikipedia. (2020, April 21). First-move advantage in chess. Retrieved from https://en.wikipedia.org/wiki/First-move_advantage_in_chess#cite_note-1

Yow, K. S. (2019). *Tutte-Whitney polynomials for directed graphs and maps* (Doctoral thesis, Monash University, Victoria, Australia. Retrieved from https://bridges.monash.edu/articles/thesis/Tutte-Whitney_Polynomials_for_Directed_Graphs_and_Maps/7610882

Yow, K. S., Farr, G. E., & Morgan, K. J. (2018). Tutte invariants for alternating dimaps (Preprint). Retrieved from https://arxiv.org/abs/1803.05539

Appendix

A = [7, 7, 7, 7, 7, 7, 7, 0, 7, 7, 7, 7, 7, 7, 7]
A = [7, 7, 7, 7, 7, 7, 0, 1, 8, 8, 8, 8, 8, 8, 7]
A = [8, 8, 8, 8, 8, 8, 1, 1, 8, 8, 8, 8, 8, 0, 8]
B = [0, 8, 0, 9, 9, 1, 9, 1, 9, 9, 9, 8, 8, 8, 0]
B = [1, 9, 1, 10, 10, 1, 9, 1, 9, 9, 0, 9, 9, 9, 1]
B = [1, 9, 1, 10, 0, 2, 10, 2, 10, 10, 1, 10, 10, 10, 2]
B = [2, 10, 1, 10, 0, 2, 10, 2, 10, 10, 1, 10, 10, 10, 0]
B = [2, 0, 2, 11, 1, 3, 11, 3, 11, 11, 2, 11, 10, 10, 0]
B = [3, 1, 3, 12, 2, 4, 12, 4, 11, 11, 2, 0, 11, 11, 1]
B = [4, 2, 4, 13, 2, 4, 0, 5, 12, 12, 3, 1, 12, 12, 2]
B = [5, 3, 4, 0, 3, 5, 1, 6, 13, 13, 4, 2, 13, 13, 3]
B = [5, 0, 5, 1, 4, 5, 1, 6, 13, 13, 4, 2, 13, 13, 3]
B = [5, 0, 5, 1, 0, 6, 2, 7, 14, 13, 4, 2, 13, 13, 3]
B = [6, 1, 6, 2, 1, 7, 3, 8, 0, 14, 5, 3, 14, 14, 4]
B = [6, 1, 6, 2, 0, 8, 3, 8, 0, 14, 5, 3, 14, 14, 4]
B = [6, 1, 6, 2, 0, 0, 4, 9, 1, 15, 6, 4, 15, 15, 4]
B = [7, 2, 7, 3, 1, 1, 5, 10, 2, 16, 7, 5, 16, 1, 5]
A = [2, 16, 0, 6, 17, 2, 6, 11, 8, 3, 7, 3, 1, 1, 5]
A = [2, 16, 0, 6, 17, 2, 6, 11, 8, 0, 8, 4, 2, 1, 5]
A = [2, 16, 0, 6, 17, 2, 6, 11, 8, 0, 8, 4, 0, 2, 6]
A = [3, 17, 1, 7, 18, 3, 6, 11, 8, 0, 8, 4, 0, 2, 0]
A = [3, 17, 1, 7, 18, 0, 7, 12, 9, 0, 8, 4, 0, 2, 0]
A = [4, 18, 2, 7, 18, 0, 7, 12, 0, 1, 9, 5, 1, 3, 1]
A = [4, 18, 0, 8, 19, 0, 7, 12, 0, 1, 9, 5, 1, 3, 1]
A = [5, 19, 1, 9, 1, 2, 9, 14, 2, 2, 10, 6, 2, 4, 2]
A = [5, 19, 1, 9, 1, 2, 9, 14, 0, 3, 11, 6, 2, 4, 2]
A = [6, 20, 2, 10, 2, 3, 10, 14, 0, 3, 0, 7, 3, 5, 3]
A = [7, 21, 2, 10, 2, 3, 0, 15, 1, 4, 1, 8, 4, 6, 4]
A = [8, 1, 4, 12, 4, 5, 2, 17, 2, 5, 2, 9, 5, 7, 5]
A = [9, 1, 4, 0, 5, 6, 3, 18, 3, 6, 3, 10, 6, 8, 6]
A = [0, 2, 5, 1, 6, 7, 4, 19, 4, 7, 3, 10, 6, 8, 6]
A = [1, 3, 5, 1, 6, 7, 4, 19, 4, 0, 4, 11, 7, 9, 7]
A = [1, 0, 6, 2, 7, 7, 4, 19, 4, 0, 4, 11, 7, 9, 7]
A = [1, 0, 6, 2, 0, 8, 5, 20, 5, 1, 5, 12, 7, 9, 7]
A = [2, 1, 7, 3, 1, 9, 6, 21, 6, 1, 5, 0, 8, 10, 8]
A = [2, 1, 7, 3, 1, 9, 6, 21, 0, 2, 6, 1, 9, 11, 9]
A = [3, 2, 8, 4, 2, 10, 7, 22, 1, 2, 6, 1, 9, 11, 0]
B = [1, 0, 7, 2, 9, 11, 0, 10, 3, 2, 8, 4, 2, 10, 7]
B = [1, 0, 7, 0, 10, 12, 0, 10, 3, 2, 8, 4, 2, 10, 7]
B = [2, 1, 8, 0, 10, 0, 1, 11, 4, 3, 9, 5, 3, 11, 8]
B = [2, 1, 0, 1, 11, 1, 2, 12, 5, 4, 10, 5, 3, 11, 8]
B = [3, 2, 1, 2, 12, 2, 2, 12, 5, 4, 0, 6, 4, 12, 9]
B = [3, 2, 1, 2, 12, 0, 3, 13, 5, 4, 0, 6, 4, 12, 9]
B = [3, 0, 2, 3, 12, 0, 3, 13, 5, 4, 0, 6, 4, 12, 9]
B = [3, 0, 2, 0, 13, 1, 4, 13, 5, 4, 0, 6, 4, 12, 9]
B = [3, 0, 2, 0, 13, 1, 0, 14, 6, 5, 1, 6, 4, 12, 9]
A = [6, 5, 1, 0, 5, 13, 10, 23, 4, 1, 2, 0, 13, 1, 0]
B = [4, 1, 0, 1, 14, 1, 0, 14, 6, 5, 1, 0, 5, 13, 10]
B = [5, 2, 1, 2, 0, 2, 1, 15, 7, 6, 2, 1, 6, 14, 11]
B = [5, 2, 1, 0, 1, 3, 1, 15, 7, 6, 2, 1, 6, 14, 11]
B = [5, 2, 1, 0, 1, 0, 2, 16, 8, 6, 2, 1, 6, 14, 11]
B = [6, 3, 1, 0, 1, 0, 2, 16, 0, 7, 3, 2, 7, 15, 12]
B = [6, 0, 2, 1, 2, 0, 2, 16, 0, 7, 3, 2, 7, 15, 12]
B = [6, 0, 2, 1, 0, 1, 3, 16, 0, 7, 3, 2, 7, 15, 12]
B = [6, 0, 2, 1, 0, 1, 0, 17, 1, 8, 3, 2, 7, 15, 12]
B = [7, 1, 3, 1, 0, 1, 0, 17, 1, 0, 4, 3, 8, 16, 13]
B = [7, 1, 0, 2, 1, 2, 0, 17, 1, 0, 4, 3, 8, 16, 13]
B = [7, 1, 0, 2, 1, 0, 1, 18, 1, 0, 4, 3, 8, 16, 13]
B = [7, 1, 0, 0, 2, 1, 1, 18, 1, 0, 4, 3, 8, 16, 13]
A = [1, 0, 4, 0, 9, 17, 14, 23, 7, 1, 0, 0, 2, 1, 1]
A = [2, 1, 5, 1, 10, 18, 0, 24, 8, 2, 1, 1, 3, 2, 2]
A = [3, 2, 6, 2, 11, 1, 2, 26, 10, 3, 2, 2, 4, 3, 3]
A = [4, 3, 7, 3, 11, 1, 2, 26, 0, 4, 3, 3, 5, 4, 4]
A = [4, 3, 7, 0, 12, 2, 3, 26, 0, 4, 3, 3, 5, 4, 4]
A = [4, 3, 7, 0, 12, 2, 0, 27, 1, 5, 3, 3, 5, 4, 4]
A = [4, 3, 7, 0, 12, 2, 0, 27, 1, 0, 4, 4, 6, 5, 5]
A = [5, 4, 8, 1, 13, 2, 0, 27, 1, 0, 4, 4, 6, 5, 0]
A = [6, 5, 9, 1, 0, 3, 1, 28, 2, 1, 5, 5, 7, 6, 1]
A = [6, 5, 0, 2, 1, 4, 2, 29, 3, 2, 6, 6, 7, 6, 1]
A = [7, 6, 1, 2, 1, 4, 2, 29, 3, 2, 6, 0, 8, 7, 2]
B = [0, 3, 7, 1, 0, 7, 2, 18, 7, 6, 0, 2, 1, 4, 2]
A = [7, 6, 0, 0, 1, 4, 0, 39, 1, 3, 7, 0, 0, 7, 2]
B = [1, 3, 0, 1, 1, 8, 3, 22, 8, 7, 0, 0, 1, 4, 0]
B = [2, 4, 0, 1, 1, 8, 3, 22, 8, 0, 1, 1, 2, 5, 1]
B = [2, 0, 1, 2, 2, 9, 3, 22, 8, 0, 1, 1, 2, 5, 1]
B = [2, 0, 1, 2, 2, 0, 4, 23, 9, 1, 2, 2, 3, 6, 2]
B = [3, 1, 1, 2, 2, 0, 4, 23, 9, 1, 2, 2, 3, 6, 0]
A = [9, 1, 0, 3, 4, 0, 0, 39, 3, 0, 1, 2, 2, 0, 4]
A = [9, 1, 0, 3, 0, 1, 1, 40, 4, 0, 1, 2, 2, 0, 4]
A = [9, 1, 0, 3, 0, 1, 1, 40, 0, 1, 2, 3, 3, 0, 4]
A = [10, 1, 0, 3, 0, 1, 1, 40, 0, 1, 2, 3, 0, 1, 5]
A = [0, 2, 1, 4, 1, 2, 2, 41, 1, 2, 3, 3, 0, 1, 5]
A = [0, 2, 1, 4, 1, 2, 2, 41, 1, 2, 0, 4, 1, 2, 5]
A = [1, 2, 1, 4, 1, 2, 2, 41, 1, 2, 0, 4, 1, 0, 6]
B = [1, 0, 1, 5, 1, 0, 0, 30, 0, 2, 1, 4, 1, 2, 2]
B = [1, 0, 1, 0, 2, 1, 1, 31, 1, 2, 1, 4, 1, 2, 2]
A = [1, 2, 1, 4, 1, 0, 3, 49, 1, 0, 1, 0, 2, 1, 1]
A = [0, 3, 1, 4, 1, 0, 3, 49, 1, 0, 1, 0, 2, 1, 1]
A = [0, 0, 2, 5, 2, 0, 3, 49, 1, 0, 1, 0, 2, 1, 1]
A = [0, 0, 2, 5, 0, 1, 4, 49, 1, 0, 1, 0, 2, 1, 1]
A = [0, 0, 2, 5, 0, 1, 0, 50, 2, 1, 2, 0, 2, 1, 1]
A = [0, 0, 2, 5, 0, 1, 0, 50, 2, 1, 0, 1, 3, 1, 1]
A = [1, 0, 2, 5, 0, 1, 0, 50, 2, 1, 0, 1, 0, 2, 2]
B = [2, 1, 0, 0, 1, 2, 0, 31, 0, 0, 2, 5, 0, 1, 0]

A = [0, 0, 0, 5, 0, 0, 1, 53, 2, 1, 0, 0, 0, 2, 0]
B = [0, 0, 1, 0, 0, 2, 0, 34, 0, 0, 0, 5, 0, 0, 0]
A = [0, 0, 0, 0, 1, 1, 1, 57, 1, 0, 1, 0, 0, 2, 0]
B = [1, 0, 0, 1, 0, 2, 0, 34, 0, 0, 0, 0, 1, 1, 1]
A = [0, 0, 0, 0, 0, 2, 1, 57, 1, 0, 0, 1, 0, 2, 0]
A = [0, 0, 0, 0, 0, 0, 2, 58, 1, 0, 0, 1, 0, 2, 0]
A = [0, 0, 0, 0, 0, 0, 0, 59, 2, 0, 0, 1, 0, 2, 0]
A = [0, 0, 0, 0, 0, 0, 0, 59, 0, 1, 1, 1, 0, 2, 0]
B = [0, 0, 2, 1, 0, 2, 0, 34, 0, 0, 0, 0, 0, 0, 0]
B = [0, 0, 0, 2, 1, 2, 0, 34, 0, 0, 0, 0, 0, 0, 0]
B = [0, 0, 0, 2, 0, 3, 0, 34, 0, 0, 0, 0, 0, 0, 0]
B = [0, 0, 0, 2, 0, 0, 1, 35, 1, 0, 0, 0, 0, 0, 0]
A = [0, 1, 0, 0, 0, 0, 0, 59, 0, 0, 0, 2, 0, 0, 1]
B = [0, 0, 0, 2, 0, 0, 0, 36, 0, 1, 0, 0, 0, 0, 0]
B = [0, 0, 0, 0, 1, 1, 0, 36, 0, 1, 0, 0, 0, 0, 0]
B = [0, 0, 0, 0, 0, 1, 0, 38, 0, 0, 0, 0, 0, 0, 0]
B = [0, 0, 0, 0, 0, 0, 1, 38, 0, 0, 0, 0, 0, 0, 0]
B = [0, 0, 0, 0, 0, 0, 0, 39, 0, 0, 0, 0, 0, 0, 0]
Player A owns 59 stones and Player B owns 39 stones.
Player A is the winner.