CHOPSTICKS

Members:

Foo Gesar	**2A1 (10) (Leader)**
Aidan Ong	**2A3 (01)**
Oh Li-Wei	**2A3 (21)**
Sa Ziyang	**2A3 (24)**

Mentor:

Mr. Chen Zenghui

CONTENTS

page

1. Introduction 03
    - 1.1. What is Chopsticks 03
    - 1.2. Terminology 03
    - 1.3. Rationale 03
2. Literature Review 04
3. Objectives and Research Question 04
4. Methodology 05
5. Results 06
    - 5.1. Notation used 06
    - 5.2. Lemmas 06
        - 5.2.1. Lemma 1 06
        - 5.2.2. Lemma 2 07
        - 5.2.3. Lemma 3 08
        - 5.2.4. Lemma 4 09
    - 5.3. Research Question 1 10
    - 5.4. Research Question 2 11
    - 5.5. Research Question 3 13
6. Conclusion 13
7. Further Extension 13
8. References 14
9. Appendix 15

1 INTRODUCTION

1.1 WHAT IS CHOPSTICKS

Chopsticks is a hand game for two players in which players extend a number of fingers from each hand and transfer those scores by taking turns to tap one hand against another. It has roots in Japan, and has been called Finger Chess, Swords, Split, Magic Fingers, Chinese Fingers, Cherries, Sticks, and other names.

To play the game, each player starts with 1 "chopstick" in each hand and will take turns to tap or transfer. When both of the opponent's hands are killed, the player wins.

1.2 TERMINOLOGY

Terms	Definition
Chopsticks	Each finger that you put out represents a 'chopstick'.
Chopsticks Limit (*n*)	The amount of 'chopsticks' you have to have in each hand before the hand is 'dead'.
Tap	Use your hand to hit another player's hand where their 'chopsticks' number on the opponent's hand is the sum of the 'chopsticks' on your hand you used to tap and the amount of 'chopsticks' on the opponent's hand that is being tapped. E.g.: Hand used to tap: 2 'chopsticks' Opponent's hand before: 1 'chopsticks' Opponent's hand after: 1 + 2 = 3 'chopsticks'
Transfer	Changing the numbers of 'chopsticks' between your hands so that they are different as before but still have the same sum. E.g.: Your hands before: 2 'chopsticks' and 3 'chopsticks' respectively Your hands after: 1 'chopsticks' and 4 'chopsticks' respectively Not allowed: 3 'chopsticks' and 2 'chopsticks' respectively
Kill	A hand is killed when the amount of chopsticks in the hand is larger than or equal to the chopstick limit.

1.3 RATIONALE

This is a game we have played since we were young. We hope to gain a broader and deeper insight of the game, and also find out the strategy to ensure victory in the game. Hence, we have decided to embark on this project.

2 LITERATURE REVIEW

Chopstick is a game originated from Japan. Up till today, not much research has been conducted on this topic. Most of the research conducted on this topic is focused mainly on the winning strategies of the original chopsticks game.

In the *World Heritage Encyclopedia,* it introduces the Chopsticks game and shows us different types of variations that different people have used and played, such as ‘exact play’, ‘knub’, ‘quarters’ and ‘suicidal’.

Besides, an article on *Kipkis* (2018) shows us a general idea of the winning strategy of the Chopsticks game as well as some details and some explanation of the winning strategy. However, it only states the winning strategy for the original game where there are 2 players, each having 2 hands and the chopstick limit is 5, while we will also be investigating the winning strategy for a game with a varied chopstick limit.

We have found that the game, Nim is very similar to Chopsticks. They both involve sticks in piles. *Marianne (2019)* explains more about nims and its solution. We hope to use the tactics employed in solving Nim to Chopsticks. These tactics include invariants and binary conversion.

3 OBJECTIVES AND RESEARCH QUESTIONS

This project aims to achieve the following objectives:

- To investigate the winning strategy for the original chopsticks game.
- To investigate some variations of the game.
- To prove our results, mathematically or computationally.

Hence, 3 research questions were proposed:

1) What is the winning strategy for the original chopsticks game (chopsticks limit is 5)?
2) Will there be a winning strategy if we vary the chopstick limit?
3) How do we prove or disprove our results from Research Question 2, mathematically, or computationally?

4 METHODOLOGY

Firstly, we researched and explored the Chopsticks game through tactics such as listing and brute-forcing, and tried to find patterns in our data. Secondly, we learnt combinatorial approaches that helped to solve the question. And lastly, we proved our formula by methods such as induction or C++.

5 RESULTS

5.1 NOTATION USED

This notation would be used throughout the paper, and would indicate a game state at a given time.

$$\begin{array}{llllll} \text{P1 fingers} & \rightarrow & 1 & 3 & \bullet & \leftarrow \quad \text{(point indicates player's turn)} \\ \text{P2 fingers} & \rightarrow & 2 & 6 & & \end{array}$$

5.2 LEMMAS

Four lemmas were identified to assist in finding the strategies.

5.2.1 LEMMA 1

$$\boxed{\begin{matrix}0 & 1 & \\ 0 & x & \bullet\end{matrix} \longrightarrow \begin{matrix}0 & 1 & \bullet \\ 0 & x+1 & \end{matrix} \quad (x+1 < n)}$$

Proof:

$$\begin{tikzcd}
\begin{matrix}0 & 1 & \\ 0 & x & \bullet\end{matrix} \arrow[r] & \begin{matrix}0 & 1 & \bullet \\ 1 & x-1 & \end{matrix} \arrow[r] \arrow[ddr] & \begin{matrix}0 & 1 & \\ 2 & x-1 & \bullet\end{matrix} \arrow[r] & \begin{matrix}0 & 1 & \bullet \\ 0 & x+1 & \end{matrix} \\
 & & \text{(or)} & \\
 & & \begin{matrix}0 & 1 & \\ 1 & x & \bullet\end{matrix} \arrow[uur] &
\end{tikzcd}$$

5.2.2 LEMMA 2

When $\begin{smallmatrix} x & y & \\ n\text{-}3 & n\text{-}2 & \bullet \end{smallmatrix}$, if x is odd and y is even, P2 wins.

Proof:

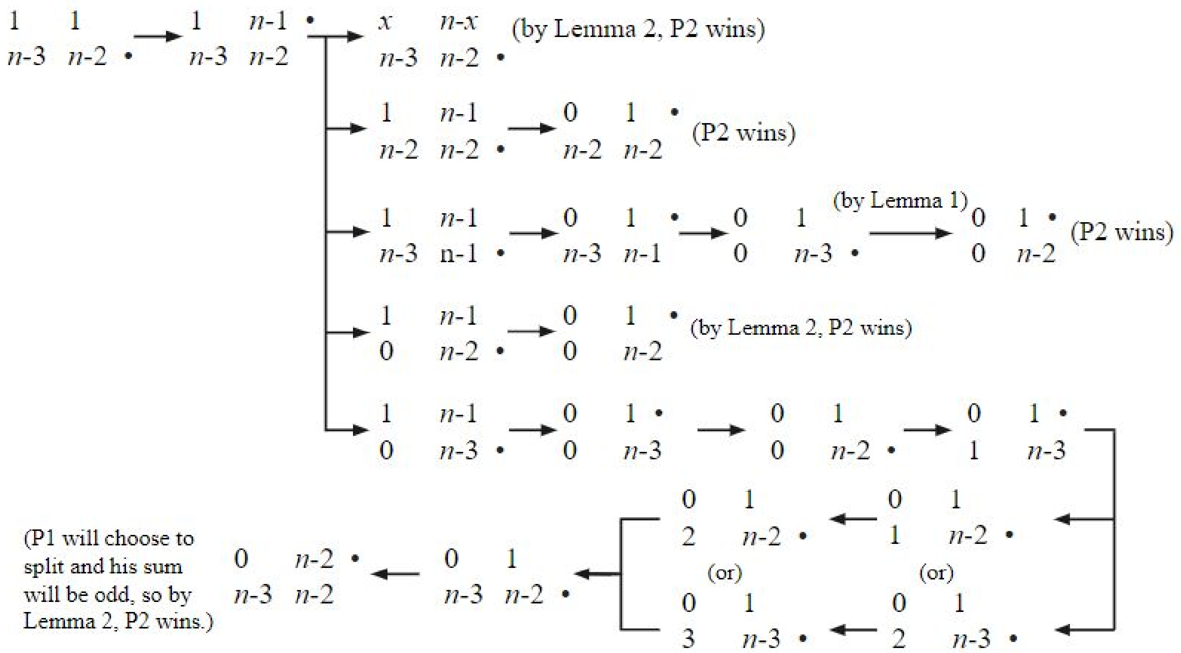
$$\begin{tikzcd}
\begin{smallmatrix} x & y & \\ n\text{-}3 & n\text{-}2 & \bullet \end{smallmatrix} \arrow[r] & \begin{smallmatrix} 0 & x & \bullet \\ n\text{-}3 & n\text{-}2 & \end{smallmatrix} \arrow[d] & & & \\
& \begin{smallmatrix} x_1 & y_1 & \\ n\text{-}3 & n\text{-}2 & \bullet \end{smallmatrix} \arrow[d] & \text{(x will be split since if P1 attacks he will lose, and note that either x_1 or y_1 is even since $x\equiv 1 \pmod 2$)} & & \\
& \begin{smallmatrix} x_1 & 0 & \bullet \\ n\text{-}3 & n\text{-}2 & \end{smallmatrix} \arrow[d, dotted, "\text{(continue…)}"] & \text{(WLOG, assume $y_1 \equiv 0 \pmod 2$)} & & \\
& \begin{smallmatrix} 0 & 1 & \bullet \\ n\text{-}3 & n\text{-}2 & \end{smallmatrix} \arrow[r] & \begin{smallmatrix} 0 & 1 & \\ n\text{-}2 & n\text{-}2 & \bullet \end{smallmatrix} \arrow[r] & \begin{smallmatrix} 0 & 1 & \bullet \\ n\text{-}3 & n\text{-}1 & \end{smallmatrix} \arrow[r] & \begin{smallmatrix} 0 & 1 & \\ 0 & n\text{-}3 & \bullet \end{smallmatrix} \arrow[d, "\text{(By Lemma 1)}"'] \\
& \text{(P2 wins)} & \begin{smallmatrix} 0 & 0 & \bullet \\ 0 & n\text{-}1 & \end{smallmatrix} & \begin{smallmatrix} 0 & 1 & \\ 0 & n\text{-}1 & \bullet \end{smallmatrix} \arrow[l] & \begin{smallmatrix} 0 & 1 & \bullet \\ 0 & n\text{-}2 & \end{smallmatrix} \arrow[l]
\end{tikzcd}$$

5.2.3 LEMMA 3

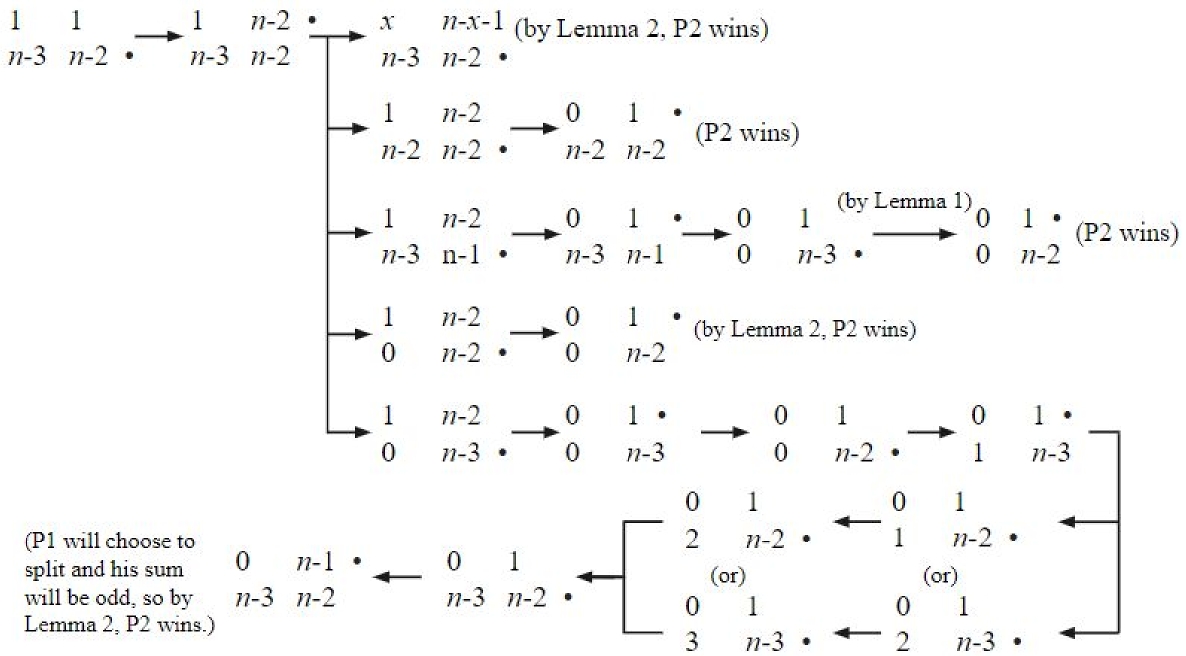
$$\boxed{\text{When } \begin{matrix}1 & 1 & \\ n\text{-}3 & n\text{-}2 & \bullet\end{matrix}\text{, and } n \geq 5\text{, P2 wins.}}$$

Proof:

When n is odd,



When n is even,



5.2.4 LEMMA 4

When $n \geq 8$, $\begin{matrix}1 & 1 & \bullet\\ 1 & 1 & \end{matrix} \longrightarrow \begin{matrix}1 & 1 & \\ n\text{-}3 & n\text{-}2 & \bullet\end{matrix}$

Proof:

Let $\begin{matrix}1 & 1 & \\ x & y & \bullet\end{matrix}$ or $\begin{matrix}0 & 2 & \\ x & y & \bullet\end{matrix}$ be the game state on P2's turn. Then,

$$\begin{tikzcd}
\begin{matrix}1 & 1 & \\ x & y & \bullet\end{matrix} \arrow[r] \arrow[dr] & \begin{matrix}1 & 1 & \bullet\\ \lceil \frac{x+y}{2} \rceil & \lceil \frac{x+y-1}{2} \rceil & \end{matrix} \\
 & \begin{matrix}1 & 1 & \bullet\\ \lceil \frac{x+y}{2} \rceil + 1 & \lceil \frac{x+y-1}{2} \rceil - 1 & \end{matrix}
\end{tikzcd}$$

$$\begin{tikzcd}
\begin{matrix}0 & 2 & \\ x & y & \bullet\end{matrix} \arrow[r] \arrow[dr] & \begin{matrix}0 & 2 & \bullet\\ \lceil \frac{x+y}{2} \rceil & \lceil \frac{x+y-1}{2} \rceil & \end{matrix} \\
 & \begin{matrix}0 & 2 & \bullet\\ \lceil \frac{x+y}{2} \rceil + 1 & \lceil \frac{x+y-1}{2} \rceil - 1 & \end{matrix}
\end{tikzcd}$$

When $\begin{matrix}0 & 2 & \\ n\text{-}3 & n\text{-}3 & \bullet\end{matrix}$ or $\begin{matrix}0 & 2 & \\ n\text{-}4 & n\text{-}3 & \bullet\end{matrix}$, P2 is unable to split without losing a hand. Strategies should be as such:

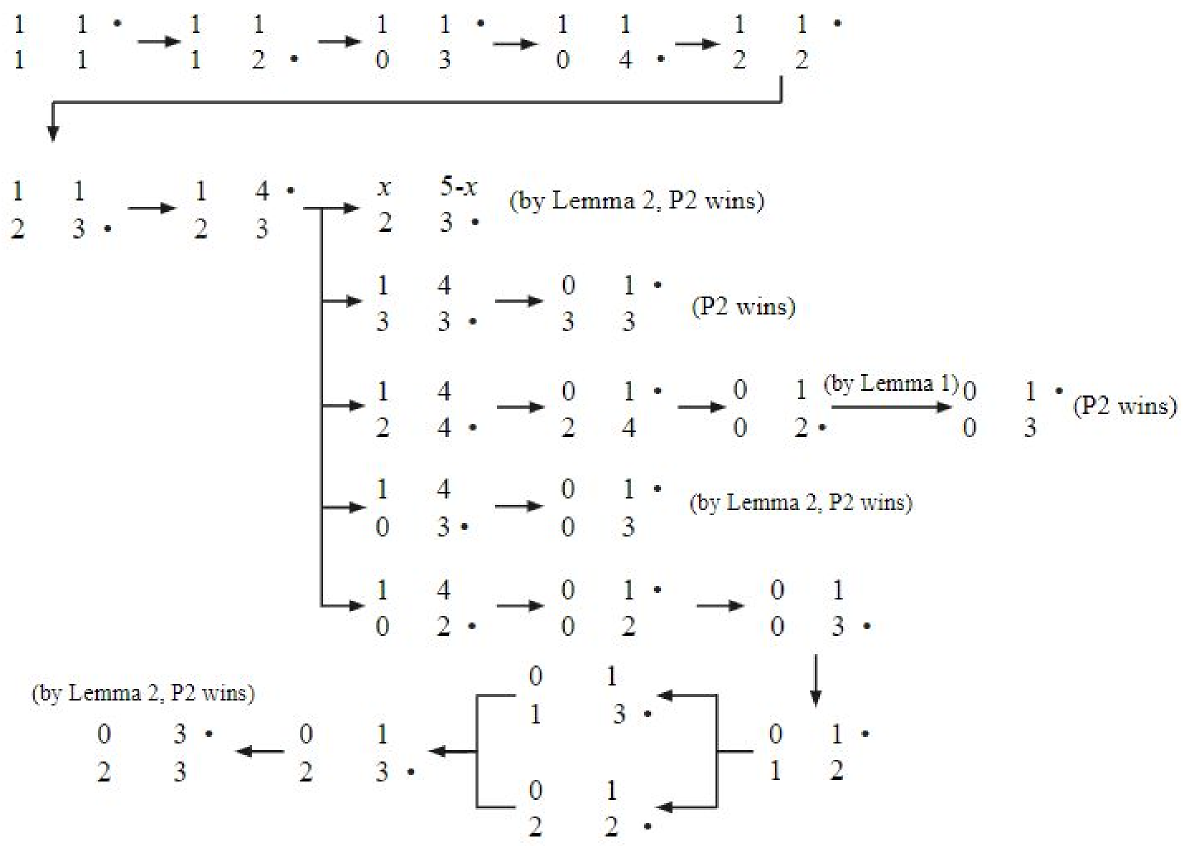
$$\begin{tikzcd}
\begin{matrix}0 & 2 & \\ n\text{-}3 & n\text{-}3 & \bullet\end{matrix} \arrow[r] & \begin{matrix}0 & 2 & \bullet\\ n\text{-}4 & n\text{-}2 & \end{matrix} \arrow[r] & \begin{matrix}0 & 2 & \\ 0 & n\text{-}4 & \bullet\end{matrix} \arrow[r] & \begin{matrix}0 & 2 & \bullet\\ \lceil \frac{n-5}{2} \rceil & \lceil \frac{n-4}{2} \rceil & \end{matrix}
\end{tikzcd}$$
(loops back to an earlier state of the game, $n \geq 6$)

$$\begin{tikzcd}
\begin{matrix}0 & 2 & \\ n\text{-}4 & n\text{-}3 & \bullet\end{matrix} \arrow[r] & \begin{matrix}0 & 2 & \bullet\\ n\text{-}6 & n\text{-}1 & \end{matrix} \arrow[r] \arrow[d] & \begin{matrix}0 & 2 & \\ 0 & n\text{-}6 & \bullet\end{matrix} \arrow[r] & \begin{matrix}0 & 2 & \bullet\\ \lceil \frac{n-7}{2} \rceil & \lceil \frac{n-6}{2} \rceil & \end{matrix} & & \\
 & \begin{matrix}1 & 1 & \\ n\text{-}6 & n\text{-}1 & \bullet\end{matrix} \arrow[d] & \begin{matrix}1 & 1 & \\ n\text{-}4 & \text{n-}2 & \bullet\end{matrix} \arrow[r] & \begin{matrix}1 & 1 & \bullet\\ n\text{-}3 & n\text{-}3 & \end{matrix} \arrow[r] & \begin{matrix}0 & 2 & \\ n\text{-}3 & n\text{-}3 & \bullet\end{matrix} & \\
 & \begin{matrix}1 & 1 & \bullet\\ n\text{-}4 & n\text{-}3 & \end{matrix} \arrow[uul] \arrow[ur] \arrow[r] & \text{(continue advancing to } \begin{matrix}1 & 1 & \\ n\text{-}3 & n\text{-}2 & \bullet\end{matrix}\text{)} & & &
\end{tikzcd}$$
(loops back to an earlier state of the game, $n \geq 8$)

5.3 RESEARCH QUESTION 1

What is the winning strategy for the original chopsticks game (i.e. 2-2-5)?

The strategy has been developed by drawing a tree, and finding the optimal path. The intuition behind this strategy is useful and important to the Research Question 2.

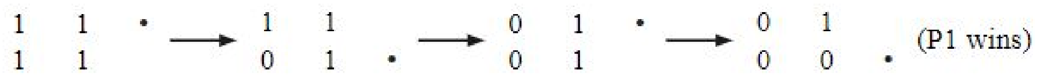


5.4 RESEARCH QUESTION 2

What is the winning strategy if we vary the chopstick limit?

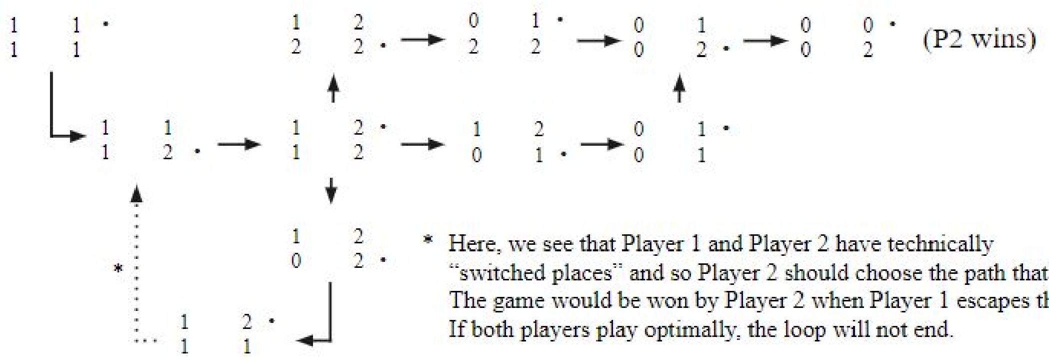
We determined the strategy for each value of *n* as shown below.

When $n = 2$,



P1 has a winning strategy.

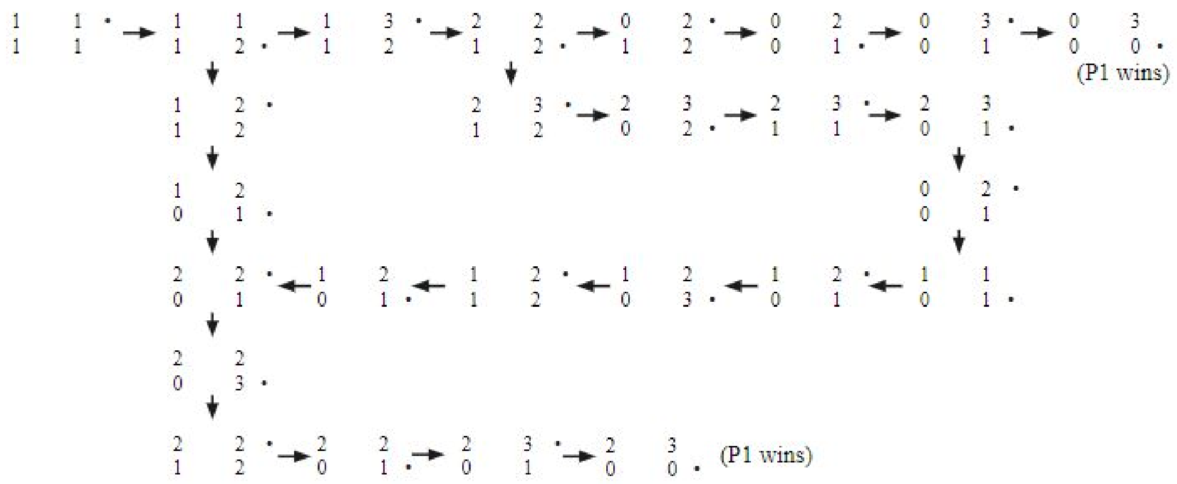
When $n = 3$,



* Here, we see that Player 1 and Player 2 have technically "switched places" and so Player 2 should choose the path that loops back. The game would be won by Player 2 when Player 1 escapes the loop. If both players play optimally, the loop will not end.

P2 has a winning strategy.

When $n = 4$,



P1 has a winning strategy.

By Lemma 3, when $n \geq 5$ and the game state is $\begin{smallmatrix} 1 & 1 & \\ n\text{-}3 & n\text{-}2 & \bullet \end{smallmatrix}$, P2 wins.

When $6 \leq n < 8$, we only need to show the solution for $\begin{smallmatrix} 0 & 2 & \\ n\text{-}4 & n\text{-}3 & \bullet \end{smallmatrix}$ as we can see in Lemma 4.

When $n = 5$, we need to show the solution for $\begin{smallmatrix} 0 & 2 & \\ n\text{-}3 & n\text{-}3 & \bullet \end{smallmatrix}$ and $\begin{smallmatrix} 0 & 2 & \\ n\text{-}4 & n\text{-}3 & \bullet \end{smallmatrix}$ as we can see in Lemma 4.

When $n = 5$,

Note that $\begin{smallmatrix} 0 & 2 & \\ 1 & 2 & \bullet \end{smallmatrix}$ can only be derived from $\begin{smallmatrix} 1 & 1 & \bullet \\ 1 & 2 & \end{smallmatrix}$, which in turn, can only be derived from $\begin{smallmatrix} 1 & 1 & \\ 1 & 1 & \bullet \end{smallmatrix}$, which should not occur given our strategy. Similarly, by backtracking, we can see that $\begin{smallmatrix} 0 & 2 & \\ 2 & 2 & \bullet \end{smallmatrix}$ is not possible either.

$$\begin{smallmatrix} 1 & 1 & \bullet \\ 1 & 1 & \end{smallmatrix} \to \begin{smallmatrix} 1 & 1 & \\ 1 & 2 & \bullet \end{smallmatrix} \to \begin{smallmatrix} 1 & 1 & \bullet \\ 0 & 3 & \end{smallmatrix} \to \begin{smallmatrix} 1 & 1 & \\ 0 & 4 & \bullet \end{smallmatrix} \to \begin{smallmatrix} 1 & 1 & \bullet \\ 2 & 2 & \end{smallmatrix} \to \begin{smallmatrix} 1 & 1 & \\ 2 & 3 & \bullet \end{smallmatrix} \text{ (By Lemma 3, P2 wins)}$$

P2 has a winning strategy.

When $n = 6$,

$$\begin{tikzcd}
\begin{smallmatrix} 0 & 2 & \\ 2 & 3 & \bullet \end{smallmatrix} \arrow[r] & \begin{smallmatrix} 0 & 5 & \bullet \\ 2 & 3 & \end{smallmatrix} \arrow[r] \arrow[d] & \begin{smallmatrix} 1 & 4 & \\ 2 & 3 & \bullet \end{smallmatrix} \arrow[r] & \begin{smallmatrix} 0 & 1 & \bullet \\ 2 & 3 & \end{smallmatrix} \arrow[r] & \begin{smallmatrix} 0 & 1 & \\ 3 & 3 & \bullet \end{smallmatrix} \arrow[r] & \begin{smallmatrix} 0 & 1 & \bullet \\ 2 & 4 & \end{smallmatrix} \arrow[r] & \begin{smallmatrix} 0 & 1 & \\ 3 & 4 & \bullet \end{smallmatrix} \arrow[r] & \begin{smallmatrix} 0 & 5 & \bullet \\ 3 & 4 & \end{smallmatrix} \\
 & \begin{smallmatrix} 2 & 3 & \\ 2 & 3 & \bullet \end{smallmatrix} \arrow[ul] & & & & & & \text{(By Lemma 2, P2 wins)}
\end{tikzcd}$$

P2 has a winning strategy.

When $n = 7$,

$$\begin{tikzcd}
\begin{smallmatrix} 0 & 2 & \\ 3 & 4 & \bullet \end{smallmatrix} \arrow[r] & \begin{smallmatrix} 0 & 5 & \bullet \\ 3 & 4 & \end{smallmatrix} \arrow[r] \arrow[d] & \begin{smallmatrix} 1 & 4 & \\ 3 & 4 & \bullet \end{smallmatrix} \arrow[r] & \begin{smallmatrix} 0 & 1 & \bullet \\ 3 & 4 & \end{smallmatrix} \arrow[r] & \begin{smallmatrix} 0 & 1 & \\ 4 & 4 & \bullet \end{smallmatrix} \arrow[r] & \begin{smallmatrix} 0 & 1 & \bullet \\ 3 & 5 & \end{smallmatrix} \arrow[r] & \begin{smallmatrix} 0 & 1 & \\ 4 & 5 & \bullet \end{smallmatrix} \arrow[r] & \begin{smallmatrix} 0 & 5 & \bullet \\ 4 & 5 & \end{smallmatrix} \\
 & \begin{smallmatrix} 2 & 3 & \\ 3 & 4 & \bullet \end{smallmatrix} \arrow[ul] & & & & & & \text{(By Lemma 2, P2 wins)}
\end{tikzcd}$$

P2 has a winning strategy.

When $n \geq 8$, by Lemma 3 and 4, P2 wins.

5.5 RESEARCH QUESTION 3

How do we prove or disprove our results from Research Question 2, mathematically or computationally?

We proved our results from Research Question 2 mathematically by using trees to exhaust all possible game states.

As a computational test, we coded a programme on C++ (refer to the code in Appendix A) to simulate the games and collect data. Our strategy has been put on the respective players. A simple AI has been put on the opponent, to simulate a new player. 10 000 000 tests have been run on each value of *n* up to 10, and shown below are the results:

Value of *n*	2	3	4	5	6	7	8	9	10
Looped games (%)	0	10.2	0	0	0.6	0.3	0.4	0.3	0.3
Success rate for P2 (%)	0	100	0	100	100	100	100	100	100
Winner	P1	P2	P1	P2	P2	P2	P2	P2	P2

6 CONCLUSION

For Research Question 1, we have found a strategy for the original game. Player 2 will win if he plays optimally.

For Research Question 2, we have found strategies for variations of the game with different chopstick limits, *n*. When $n = 2$ or $n = 4$, Player 1 will win if he plays optimally. For other values of *n*, Player 2 will win if he plays optimally.

For Research Question 3, we mathematically proved the strategy using trees, and tested the strategy using C++ code.

7 FURTHER EXTENSION

In the future, a generalisation or simplification of the current strategy would be developed. A strategy would also be found for a varied hand number or player number. Other varieties of the game such as Misère or Leftovers would also be explored.

8 REFERENCES

Japanese games – Chopsticks (hand game). (2008, October 16). Retrieved from *https://sakuraentorizasshi.wordpress.com/2008/10/16/japanese-games-chopsticks-hand-game/*

Person, & wikiHow. (2020, March 16). How to Always Win Chopsticks. Retrieved from *https://www.wikihow.com/Always-Win-Chopsticks*

Person, & wikiHow. (2019, March 29). How to Play Chopsticks. Retrieved from *https://www.wikihow.com/Play-Chopsticks*

Chopsticks Game. (n.d.). Retrieved from *https://www.activityvillage.co.uk/chopsticks-game*

9 APPENDIX

9.1 APPENDIX A

```
#include <bits/stdc++.h>
using namespace std;

int phase=1, cap, hands[4], sum, test;

void setphase(){
	if(phase==6) phase=5;
	if(hands[2]==0&&hands[3]==0) phase=-1;
	if(hands[0]==0&&hands[1]==0) phase=0;
	if(hands[0]==0&&hands[1]==1&&hands[2]+hands[3]<cap-2) phase=4;
	if(hands[0]==0&&hands[1]>=1&&(hands[2]>=cap-1||hands[3]>=cap-1)) phase=0;
	if((hands[0]+hands[1])%2==1&&hands[2]==cap-3&&hands[3]==cap-2) phase=0;
	if(phase==0||phase==3||phase==5){
		return;
	}
	if(hands[0]==0&&hands[1]==2&&hands[2]==cap-4&&hands[3]==cap-3) phase=2;
	if(hands[0]==1&&hands[1]==1&&hands[2]==cap-3&&hands[3]==cap-2&&cap!=4) phase=3;
	if(((hands[0]==1&&hands[1]==1)||(hands[0]==0&&hands[1]==2))&&(hands[2]!=cap-4||hands[3]!=ca
p-3)&&hands[2]!=0&&hands[3]!=0) phase=1;
	return;
}
void reorder(){
	int temp;
	if(hands[0]>hands[1]){
		temp=hands[0];
		hands[0]=hands[1];
		hands[1]=temp;
	}
	if(hands[2]>hands[3]){
		temp=hands[2];
		hands[2]=hands[3];
		hands[3]=temp;
	}
	return;
}
void P1(){
	if(hands[1]>=cap-hands[3]&&hands[1]<=cap-hands[2]){
		hands[3]=0;
		return;
	}
	int hit=4,split=0,choices;
	if(hands[0]==hands[1]) hit/=2;
	if(hands[2]==hands[3]) hit/=2;
	split=(hands[0]+hands[1])/2;
	srand(time(0));
	choices=rand()%(split+hit);
	if(choices<hit){
		if(hands[0]==0&&hands[1]+hands[3]<cap&&2*hands[1]+hands[3]>cap){
			if(hands[1]+hands[2]>=cap) hands[2]=0;
			else hands[2]+=hands[1];
```

```
        }
        else{
            int one, two, cnt=0;
            do{
                one=rand()%2;
                two=rand()%2+2;
            }while(hands[one]!=0&&hands[two]!=0&&cnt++<4);
            if(hands[one]+hands[two]>=cap) hands[two]=0;
            else hands[two]+=hands[one];
        }
    }
    else{
        int num, a=hands[0]+hands[1];
        do{
            num=rand()%split;
            if(num>=hands[0])num++;
            hands[0]=num;
            hands[1]=a-num;
        }while(num==0&&a-num>cap-hands[3]&&split>2);
    }
    return;
}
int main(){
    float loopcnt=0, rate=0, movesum=0;
    int movecnt;
    cin>>cap;
    if(cap<8) return 0;
    cin>>test;
    for(int i=0; i<test; i++){
        movecnt=0;
        hands[0]=1,hands[1]=1,hands[2]=1,hands[3]=1,phase=1;
        while(phase!=0&&phase!=-1&&movecnt<100000){
            P1();
            reorder();
            setphase();
            sum=hands[2]+hands[3];
            if(phase==1){
                if(hands[2]==ceil(sum-1)&&hands[3]==ceil(sum))
hands[2]=ceil(sum-1)-1,hands[3]=ceil(sum)+1;
                else hands[2]=ceil(sum-1), hands[3]=ceil(sum);
            }
            if(phase==2){
                if(hands[2]==cap-4&&hands[3]==cap-3) hands[2]=cap-6, hands[3]=cap-1;
                if(hands[2]==0&&hands[3]==cap-6)          hands[2]=ceil(cap-7),
hands[3]=ceil(cap-6);
            }
            if(phase==3){
                if(cap%2==1){
                    if(hands[2]==cap-3&&hands[3]==cap-2) hands[1]=cap-1;
                    else
if(hands[1]==cap-1&&(hands[2]>=1||hands[3]>=1))hands[1]=0;
                }
                else{
                    if(hands[2]==cap-3&&hands[3]==cap-2) hands[1]=cap-2;
                    else         if(hands[1]==cap-2&&(hands[2]>=2||hands[3]>=2))
hands[1]=0;
                }
```

```
                    if      (hands[0]==0&&hands[1]==1&&hands[2]==0&&hands[3]==cap-2)
phase=6;
                }
                if(phase==4){
                    if(sum==cap-2) hands[2]=0, hands[3]=cap-2;
                    else{
                        if(hands[2]==ceil(sum-1)&&hands[3]==ceil(sum))
hands[2]=ceil(sum-1)-1,hands[3]=ceil(sum)+1;
                        else hands[2]=ceil(sum-1), hands[3]=ceil(sum);
                    }
                }
                if(phase==5){
                    if(hands[2]==0&&hands[3]==cap-2) hands[2]=cap-3, hands[3]=1;
                    else if(hands[2]==cap-3&&hands[3]==cap-2){
                        if(cap%2==1) hands[1]==cap-2;
                        else hands[1]==cap-1;
                    }
                    else{
                        if(hands[2]==cap-2) hands[2]=cap-3, hands[3]=sum-cap+3;
                        else hands[2]=cap-2, hands[3]=sum-cap+2;
                    }
                }
                reorder();
                movecnt++;
            }
            if(phase==0) rate++;
            if(movecnt<100000) movesum+=movecnt;
            else loopcnt++;
        }
        rate/=(test-loopcnt);
        cout<<"\nSuccess rate is: "<<100*rate<<"%\nTotal moves done (loops not included): "<<movesum;
        movesum/=(test-loopcnt);
        cout<<"\nAverage moves required: "<<movesum<<"\nLooped games: "<<loopcnt/test*100<<"%";
        return 0;
}
```