IE 616 Mini Project
The Cake Cutting Problem

Guramrit Singh **Deepasha** **Anupsa Swain**
210050061 **210070025** **210010012**

Indian Institute of Technology, Bombay
April 20, 2023

Contents

1 Problem Statement

There's a cake to be equally divided among n parties. Every member has a different preference of toppings i.e., some people prefer the chocolate toppings, some prefer the cherries, some just want as large a piece as possible. How do we ensure:

1. The cake is evenly divided
2. No one is envious of any other party's piece of cake

2 Concepts Used

The two concepts of division we will be using in the cake-cutting problem are:

1. **Proportional division**, in which a resource is divided among n people with subjective evaluations, giving each person at least $\frac{1}{n}$ of the resource by his/her own subjective valuation. In this case of cake cutting, all have equal view of the quantity of cake to be divided. So everyone gets exactly $\frac{1}{n}$ piece of the cake.
2. **Envy-free division**, states that when resources are allocated among people with equal rights, each person should receive a share that is, in their eyes, at least as good as the share received by any other agent. In other words, no person should feel envy.

3 Motivation and Applications

This simple problem of fair division is not limited to our friends Alice and Bob. We see it around in homes with siblings, with partners who don't compromise on cake, even with Abraham and Lot and the land of Canaan.

History aside, the problem of fair division also has mathematical applications in Sperner's Lemma and Brouwer's Fixed-Point Theorem.

The cake cutting problem touches upon many different areas, ranging from combinatorics and graph theory, to algorithms and topology. It also has many applications in economics, game theory, and decision theory, and there has been a great deal of mathematical study on this problem over the past few decades.

Before digressing further, we now look at the solution posed for our problem.

4 Algorithm

A very commonly used algorithm for the cake division problem, especially in a two-member scenario, is the ***'I cut, you choose'*** algorithm. It looks like the following:

4.1 'I (Alice) cut, You (Bob) choose'

1. Alice first cuts the cakes into two pieces such that both the pieces are of equal value to her.
2. Bob then chooses the best piece of cake according to him.

In this way, Bob is happy since he got to pick his choice of piece. And Alice is happy since both pieces were of equal value to her anyway. Both the players are satisfied.

Let us expand this analysis to three people now: Alice, Bob, and their new neighbour, Charlie. Can the "I cut, You choose" work here?

4.2 'I cut, you choose, he choose, he cut, ...'

Haris Aziz of UNSW Australia and Simon Mackenzie of Carnegie Mellon University, USA wrote a paper in 2017 describing a bound version that applies to n players. Here, we describe a three player version of the same:

1. Alice being the first player cuts the cake into three slices that are equally valuable from her perspective. Bob and Charlie are each asked to point to their favorite slices, and if they like different slices, we're done — they each take their favorite, Alice takes the remaining slice, and everyone goes home happy.

2. If Bob and Charlie both choose the same cake slice as their favourite, Bob will be asked to remove a little portion of it in order to make the remaining cake equal in value to his second-favorite slice. The removed portion will be saved for another time. Now that Charlie has had a chance to select her favourite piece from each of the three slices, Bob will get to make his choice, but only if Charlie didn't select the trimmed slice, in which case Bob will have to accept it. The third piece goes to Alice.

3. None of the players are envious of one another at this point. Charlie is happy because she got to make the decision first, Bob is happy because he selected one of his two top choices, and Alice is delighted because he selected one of his three original works, all of which, in his opinion, are equally good.delighted because he selected one of his three original works, all of which, in his opinion, are equally good.

4. However, the trimmed portion still needs to be separated. The fact that Alice is more than just content with the cake he has so far makes it possible to divide this portion without producing additional trimmings and entering an endless cycle of trimming and choosing. Alice would not feel cheated even if the player with the trimmed slice received all of the cake that is still available for distribution because the trimmed slice plus the trimming equals one of the original slices.

5. Now, if, for instance, Charlie received the reduced slice, the algorithm continues as follows: Bob divides the trimmings into three equal pieces before giving Alice, Charlie, and Bob the opportunity to select one each. Everyone is pleased: Charlie because she got to select first, Alice because he receives a slice he prefers to Bob's, and Bob because he believes all three slices are equally good.

5 Code Implementation

The program first asks for the number of players the cake has to be divided amongst. Then it generates a random cake with two toppings: chocolate and vanilla, their respective percentages being x and $100 - x$ (where $20 \leq x \leq 80$).
The following is what happens along the code as we proceed for either two or three players.

5.1 Number of players, n = 2

The code asks for the cutting criteria among the ingredients according to player 1. Player 2 then picks the piece of their choice. As mentioned in the algorithm, both are satisfied.

For two players, the procedure is short, two-stepped and just.

5.2 Number of players, n = 3

When it comes to three players, the complexity, code and number of steps increase in volume.

It religiously follows the algorithm discussed above; assuming the 3 players are using the interface to act according to their preferences. The code goes as:

It asks player 1 (Alice) to give the cutting information for the three pieces, such that according to her, the division is equal. The code asks player 2 (Bob) and player 3 (Charlie) for their preferences in that order. If they choose different pieces, they get their preferred choice and Alice gets the remaining piece. Everyone is now happy. The code ends.

On the other hand, if they chose same piece, player 3 gets to trim that piece so that it is equivalent to his second-most preferred piece.

Now, player 2 picks his first preference of the three. If it's the same as it was before, player 3 gets the other piece, and player 1 gets the remaining piece.

If player 2 picked any other piece, in the following code, player 2 and player 3 are interchanged.

Barring the trimming, all are happy because player 2 got his preferred choice, player 3 got one of the two equal ones he wished, and player 1 doesn't envy anyone because the pieces are equal according to her.

Now we divide the trimmed piece. player 2 cuts it into 3 equal pieces according to him. And the code proceeds with asking for their preferences in the order described in the algorithm.

The code has internal checks for wrong divisions: negative numbers, invalid divisions.

Caution: Enter only the numbers 1, 2 or 3 for picking choice of cake piece.

6 Appendix - Code in Python

(GitHub link for the code)

```
import random

num_players = int(input("Type the number of players: ")) # Number of players
chocofrac = round(60*random.random() + 20)              # Percentage of chocolate in cake
vanillafrac = 100 - chocofrac                           # Percentage of vanilla in cake

print("The current cake has " + str(chocofrac) + " percentage of chocolate topping and " \
      + str(vanillafrac)+ " percentage of vanilla topping" )
print("Player 1 gets to cut the cake ...")

################################### 2 PLAYER GAME ####################################
if num_players == 2:
    piece = [[0,0], [0,0]]                              # pieces' specifications

    print("Enter piece 1 details")
    piece[0][0] = float(input("Chocolate percentage of original cake: "))
    # Sanity check
    while (piece[0][0] > chocofrac):
        print("You can not have more than what is available :) , enter again ! ")
        piece[0][0] = float(input("Chocolate percentage of original cake: "))
    piece[0][1] = float(input("Vanilla percentage of original cake: "))
    # Sanity check
    while(piece[0][1] > vanillafrac):
```

```
        print("You can not have more than what is available :) , enter again ! ")
        piece[0][1] = float(input("Vanilla percentage of original cake: "))

    # Piece 2 details
    piece[1][0] = chocofrac - piece[0][0]
    piece[1][1] = vanillafrac - piece[0][1]

    # Pieces' specifications
    print("Piece 1 conatins " + str(piece[0][0]) + "% chocolate and " + str(piece[0][1]) \
        + "% vanilla")
    print("Piece 2 contains " + str(piece[1][0]) + "% chocolate and " + str(piece[1][1]) \
        + "% vanilla")

    print("Player 2 choose your desired piece...")
    piece_play2 = int(input()) - 1
    piece_play1 = 1 - piece_play2

    # Overall cake division
    print("Player 1 gets piece number " + str(piece_play1 + 1) + " which ammounts to " \
        + str(piece[piece_play1][0]) + "% chocolate and " + str(piece[piece_play1][1]) \
        + "% vanilla")
    print("Player 2 gets piece number " + str(piece_play2 + 1) + " which ammounts to " \
        + str(piece[piece_play2][0]) + "% chocolate and " + str(piece[piece_play2][1]) \
        + "% vanilla")

################################### 3 PLAYER GAME ####################################
elif num_players == 3:
    piece = [[0,0], [0,0], [0, 0]]                          # pieces' specifications
    trim = [[0,0], [0,0], [0, 0]]                           # trimming pieces' specifications

    print("Enter piece 1 details")
    piece[0][0] = float(input("Chocolate percentage of original cake: "))
    # Sanity check
    while (piece[0][0] > chocofrac):
        print("You can not have more than what is available :) , enter again ! ")
        piece[0][0] = float(input("Chocolate percentage of original cake: "))
    piece[0][1] = float(input("Vanilla percentage of original cake: "))
    # Sanity check
    while(piece[0][1] > vanillafrac):
        print("You can not have more than what is available :) , enter again ! ")
        piece[0][1] = float(input("Vanilla percentage of original cake: "))

    print("Enter piece 2 details")
    piece[1][0] = float(input("Chocolate percentage of original cake: "))
    # Sanity check
    while (piece[1][0] > chocofrac - piece[0][0]):
        print("You can not have more than what is available :) , enter again ! ")
        piece[1][0] = float(input("Chocolate percentage of original cake: "))
```

```
piece[1][1] = float(input("Vanilla percentage of original cake: "))
# Sanity check
while(piece[1][1] > vanillafrac - piece[0][1]):
    print("You can not have more than what is available :) , enter again ! ")
    piece[1][1] = float(input("Vanilla percentage of original cake: "))

# Piece 3 details
piece[2][0] = chocofrac - piece[0][0] - piece[1][0]
piece[2][1] = vanillafrac - piece[0][1] - piece[1][1]
# Pieces specifications'
print("Piece 1 contains " + str(piece[0][0]) + "% chocolate and " + str(piece[0][1]) \
    + "% vanilla" )
print("Piece 2 contains " + str(piece[1][0]) + "% chocolate and " + str(piece[1][1]) \
    + "% vanilla" )
print("Piece 3 contains " + str(piece[2][0]) + "% chocolate and " + str(piece[2][1]) \
    + "% vanilla" )

print("Player 2 choose your desired piece...")
piece_play2 = int(input()) - 1
print("Player 3 choose your desired piece...")
piece_play3 = int(input()) - 1

# Case 1: If both chose the same
if(piece_play3 == piece_play2):
    print("Player 2 please trim your chosen piece number " + str(piece_play2 + 1) \
        + " such that it matches with your second preferred piece")
    print("Beware! If Player 3 does not choose the trimmed piece, " \
        + "then you will have to take it")

    print("Enter trimming's details")
    trim_choco = float(input("Chocolate percentage in the trimming: "))
    while (trim_choco > piece[piece_play2][0]):
        print("You can not trim more than what is available :) , enter again !")
        trim_choco = float(input("Chocolate percentage in the trimming: "))
    trim_vanilla = float(input("Vanilla percentage in the trimming: "))
    while(trim_vanilla > piece[piece_play2][1]):
        print("You can not have more than what is available :) , enter again !")
        trim_vanilla = float(input("Vanilla percentage in the trimming: "))

    # Update pieces
    piece[piece_play2][0] -= trim_choco
    piece[piece_play2][1] -= trim_vanilla

    # Pieces and trimming specifications'
    print("Piece 1 has " + str(piece[0][0]) + "% chocolate and " + str(piece[0][1]) \
        + "% vanilla")
    print("Piece 2 has " + str(piece[1][0]) + "% chocolate and " + str(piece[1][1]) \
        + "% vanilla")
```

```
print("Piece 3 has " + str(piece[2][0]) + "% chocolate and " + str(piece[2][1]) \
    + "% vanilla")
print("Trimming has " + str(trim_choco) + "% chocolate and " + str(trim_vanilla) \
    + "% vanilla")

# Player 3 to choose a piece
print ("Player 3 choose your desired piece...")
piece_play3 = int(input()) - 1
if(piece_play2 == piece_play3):
    not_trim_who = 2                                    # player to cut the trimming
    print ("Player 2 choose your desired piece...")
    piece_play2 = int(input()) - 1
else:
    not_trim_who = 3                                    # player to cut the trimming
piece_play1 = 3 - piece_play2 - piece_play3

# Partial cake division
print("Player 1 gets piece number " + str(piece_play1 + 1) + " which ammounts to " \
    + str(piece[piece_play1][0]) + "% chocolate and " + str(piece[piece_play1][1]) \
    + "% vanilla")
print("Player 2 gets piece number " + str(piece_play2 + 1) + " which ammounts to " \
    + str(piece[piece_play2][0]) + "% chocolate and " + str(piece[piece_play2][1]) \
    + "% vanilla")
print("Player 3 gets piece number " + str(piece_play3 + 1) + " which ammounts to " \
    + str(piece[piece_play3][0]) + "% chocolate and " + str(piece[piece_play3][1]) \
    + "% vanilla")

print("Now coming to the division of the trimming ...")
print("Player " + str(not_trim_who) + " gets to cut the trimming into 3 parts ...")

print("Enter trimming's piece 1 details")
trim[0][0] = float(input("Chocolate percentage of trim cake for first piece: "))
# Sanity check
while (trim[0][0] > trim_choco):
    print("You can not have more than what is available :) , enter again ! ")
    trim[0][0] = float(input("Chocolate percentage of trim cake for first piece: "))
trim[0][1] = float(input("Vanilla percentage of trim cake for first piece: "))
# Sanity check
while(trim[0][1] > trim_vanilla):
    print("You can not have more than what is available :) , enter again ! ")
    trim[0][1] = float(input("Vanilla percentage of trim cake for first piece: "))

print("Enter trimming's piece 2 details")
trim[1][0] = float(input("Chocolate percentage of trim cake for second piece: "))
# Sanity check
while (trim[1][0] > trim_choco - trim[0][0]):
    print("You can not have more than what is available :) , enter again ! ")
    trim[1][0] = float(input("Chocolate percentage of trim cake for second piece: "))
```

```
trim[1][1]= float(input("Vanilla percentage of trim cake for second piece: "))
# Sanity check
while(trim[1][1] > trim_vanilla-trim[0][1]):
    print("You can not have more than what is available :) , enter again ! ")
    trim[1][1]= float(input("Vanilla percentage of trim cake for second piece: "))

# Trimming piece 3 details
trim[2][0] = trim_choco - trim[0][0] - trim[1][0]
trim[2][1] = trim_vanilla - trim[0][1] - trim[1][1]

# Trimmings' specifications
print("Trim Piece 1 has " + str(trim[0][0]) + "% chocolate and " + str(trim[0][1]) \
    + "% vanilla" )
print("Trim Piece 2 has " + str(trim[1][0]) + "% chocolate and " + str(trim[1][1]) \
    + "% vanilla" )
print("Trim Piece 3 has " + str(trim[2][0]) + "% chocolate and " + str(trim[2][1]) \
    + "% vanilla" )

print("Player " + str(5 - not_trim_who) + " choose a piece from the above pieces...")
trim_play_2or3 = int(input()) - 1
print("Player 1 choose a piece from the remaining 2 pieces of trimming... ")
trim_play_1 = int(input()) - 1
# Sanity check
while(trim_play_1 == trim_play_2or3):
    print("NO NO, can't take the piece that has already been taken, \
        choose some other piece number !")
    trim_play_1 = int(input()) - 1

if not_trim_who == 2:
    trim_play_3 = trim_play_2or3                    # Player 3 got to choose first
    trim_play_2 = 3 - trim_play_1 - trim_play_3
else:
    trim_play_2 = trim_play_2or3                    # Player 2 got to choose first
    trim_play_3 = 3 - trim_play_1 - trim_play_2

# Overall cake division
print("Player 1 gets piece number " + str(piece_play1 + 1) + " and trimming piece" \
    + " number " + str(trim_play_1 + 1) + " which ammounts to " \
    + str(piece[piece_play1][0] + trim[trim_play_1][0]) + "% chocolate and " \
    + str(piece[piece_play1][1] + trim[trim_play_1][1]) + "% vanilla")
print("Player 2 gets piece number " + str(piece_play2 + 1) + " and trimming piece" \
    + " number " + str(trim_play_2 + 1) + " which ammounts to " \
    + str(piece[piece_play2][0] + trim[trim_play_2][0]) + "% chocolate and " \
    + str(piece[piece_play2][1] + trim[trim_play_2][1]) + "% vanilla")
print("Player 3 gets piece number " + str(piece_play3 + 1) + " and trimming piece" \
    + " number " + str(trim_play_3 + 1) + " which ammounts to " \
    + str(piece[piece_play3][0] + trim[trim_play_3][0]) + "% chocolate and " \
    + str(piece[piece_play3][1] + trim[trim_play_3][1]) + "% vanilla")
```

```
    # Case 2: If both chose different, then we are done. Allot them their chosen pieces
    else:
        piece_play1 = 3 - piece_play2 - piece_play3
        # Overall cake division
        print("Player 1 gets piece number " + str(piece_play1 + 1) + " which ammounts to " \
            + str(piece[piece_play1][0]) + "% chocolate and " + str(piece[piece_play1][1]) \
            + "% vanilla")
        print("Player 2 gets piece number " + str(piece_play2 + 1) + " which ammounts to " \
            + str(piece[piece_play2][0]) + "% chocolate and " + str(piece[piece_play2][1]) \
            + "% vanilla")
        print("Player 3 gets piece number " + str(piece_play3 + 1) + " which ammounts to " \
            + str(piece[piece_play3][0]) + "% chocolate and " + str(piece[piece_play3][1]) \
            + "% vanilla")
else:
    print("Number of players can be either 2 or 3 !")
    exit()
print("Enjoy !")
```

7 Bibliography

The following references were used for understanding the topic of the mini-project better and for aiding the algorithm.

1. Fair Cake-Cutting, Wikipedia
2. Paper on n-player algorithm for cake-cutting
3. Fair Division, Brilliant.org