MONTY HALL PROBLEM AND BIRTHDAY PARADOX

A PROJECT REPORT

SUBMITTED IN COMPLETE FULFILMENT OF THE REQUIREMENTS

FOR THE AWARD OF DEGREE

OF

BACHELOR OF TECHNOLOGY

IN

COMPUTER ENGINEERING

SUBMITTED BY:

Pranjal Srivastava

(2K20/B5/59)

Pragati Chaudhary

(2K20/B5/56)

UNDER THE SUPERVISION OF:

Prof. S. SIVA Prasad

Ms. Pooja Yadav

COMPUTER ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(FORMERLY DELHI COLLEGE OF ENGINEERING)

BAWANA ROAD, DELHI – 110042

JULY 2021

DELHI TECHNOLOGICAL UNIVERSITY

(FORMERLY DELHI COLLEGE OF ENGINEERING)

BAWANA ROAD, DELHI – 110042

CANDIDATE'S DECLARATION

We, (Pranjal Srivastava (2K20/B5/59) and Pragati Chaudhary (2K20/B5/56)) students of Bachelor of Technology (COMPUTER ENGINEERING) hereby declare that the dissertation titled "**Monty Hall Problem and Birthday Paradox**" which is submitted by us to the Department of Applied Mathematics, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology, is original and is not copied from any source without proper citation. This work has not previously formed the basis of any award of any degree, diploma associateship, fellowship or any other similar title or recognition.

Place: Delhi — Pranjal Srivastava (2K20/B5/59)

Date:11/07/21 — Pragati Chaudhary (2K20/B5/56)

DELHI TECHNOLOGICAL UNIVERSITY

(FORMERLY DELHI COLLEGE OF ENGINEERING)

BAWANA ROAD, DELHI – 110042

CERTIFICATE

I, hereby certify that the project dissertation named "**Monty Hall Problem and Birthday Paradox**", which is submitted by Pranjal Srivastava (2K20/B5/59) and Pragati Chaudhary (2K20/B5/56) (Computer Engineering), Delhi Technological University, Delhi in complete fulfilment of the requirement for the award of the degree of the Bachelor of Technology, is a record of the project work carried out by the students under my supervision. To the best of my knowledge, this work has not been submitted in part or full for any degree or diploma to this University or elsewhere.

Place: Delhi

Prof. S. Siva Prasad

Ms. Pooja Yadav

Date:11/07/21

(Mentor)

DELHI TECHNOLOGICAL UNIVERSITY
(FORMERLY DELHI COLLEGE OF ENGINEERING)
BAWANA ROAD, DELHI – 110042

ABSTRACT

This report is a study based on two famous experiments conducted in the field of probability. The two experiments are the "**Monty Hall Problem" and the "Birthday Paradox**". Although the two problems are unrelated to much extent but they have a similarity that both are very counterintuitive to the human brain. Both the problems have been under discussion for a very long time, baffling even the most famous of the mathematicians. The problems are really easy to understand but the result may seem confusing to many. In this report, we have first of all studied the problems and their proofs. After that we also have created simulations for both the problems in python language which have really seemed to verify the result that we expected mathematically. We have also tried to discuss why the problems might seem absurd to the human brain. Overall, our aim for this project was to learn about two probability experiments in depth and simulate the experiments to some extent which we were able to achieve.

DELHI TECHNOLOGICAL UNIVERSITY

(FORMERLY DELHI COLLEGE OF ENGINEERING)

BAWANA ROAD, DELHI – 110042

ACKNOWLEDGEMENT

In performing our major project, we had to take the help and guidance of some respected people, who deserve our greatest gratitude. The completion of this assignment gives us much pleasure. We would like to show our gratitude towards **Prof . S. Siva Prasad**, our mentor for the project, who gave us a good guideline for the report throughout through numerous consultations and for always being there to motivate us and enlighten us with his profound knowledge of the subject and suggesting us improvements to the projects. We would also like to extend our deepest gratitude towards everyone who have directly and indirectly helped us to complete our project.

Many people, especially, our classmates, and team members themselves have made valuable comments and suggestions on this proposal which gave us an inspiration to improve our project. We thank all the people for their help directly and indirectly to complete our assignment.

In addition, we would like to thank Department of Applied Mathematics, Delhi Technological University for giving us the opportunity to work on this topic.

CONTENTS

INTRODUCTION TO THE “MONTY HALL PROBLEM”

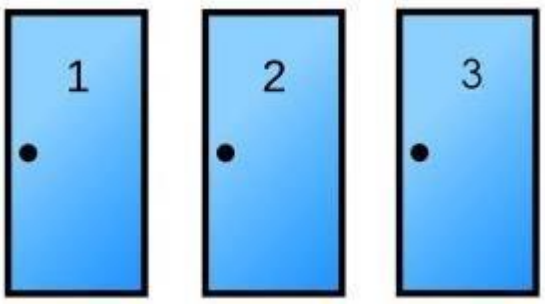
The Monty Hall Problem is a brain teaser in the form of a probability puzzle which is loosely based on an American television show called, “Let’s Make a Deal” and is named after the host Monty Hall. The problem has already been posed and solved in 1975 by a mathematician, Steve Selvin but has continued to baffle people ever since.

Let us begin by first of all understanding what the problem really is:

We would like to first of all quote the question from a reader's letter quoted in Marilyn Vos Savant's "Ask Marilyn" column in Parade magazine in 1990 which people say popularised this problem.

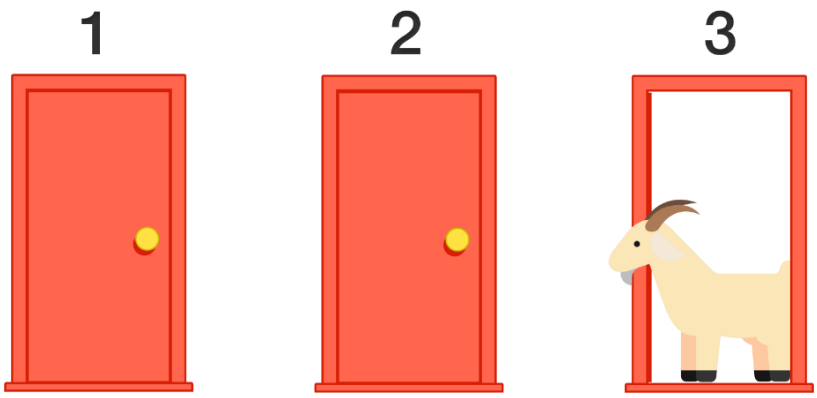
“Suppose you're on a game show, and you're given the choice of three doors: Behind one door is a car; behind the others, goats. You pick a door, say No. 1, and the host, who knows what's behind the doors, opens another door, say No. 3, which has a goat. He then says to you, "Do you want to pick door No. 2?" Is it to your advantage to switch your choice?”

Now let us try to simplify the problem:



You are on a game show and are informed that behind two of these doors are “goats” (the losing prize) and behind one of them is a “car” (the winning prize”). The probability of the

car being behind each door is equal, that is, no door is more likely to have a car or goat. Now the host gives you a chance to select one of the doors and you obviously want to try to choose the door which has the car behind it. It may or may not happen and is based on pure luck. Now comes the tricky part, the host is aware that which door has a car behind it and wants to help you a bit. The host then goes on to give you a piece of information. He opens one of the doors which has a goat behind it (obviously he will not open the door with car as it would simply give away its location and then you obviously know which door has a car behind it).



Now the host after revealing one of the goats gives you the chance to **either stick with your initial choice or switch** and choose the remaining one door that you didn't choose earlier and the also not the one which was revealed by the host (obviously you wouldn't want to choose the goat as the prize). So, the question that finally appears in front of the player is that **should you stick with your initial decision or switch to the other door to increase your chances of winning or doing either of the two doesn't matter?** (that is, there is equal probability of winning the car by switching or not switching).

THE SOLUTION TO THE "MONTY HALL PROBLEM" AND PROOF

If you are not already aware of the solution to this problem and thought of a solution, you may have in all likeliness thought that you should either stick with your decision or you might have thought that it doesn't matter, that is, switching or not switching gives you equal chance to win the car. Both of these solutions seem intuitive to the human mind specially the second one. **Out of 228 subjects in one study, only 13% chose to switch.**

However, mathematically, SWITCHING GIVES YOU **TWICE** THE ODDS TO WIN.

In simpler terms, in order to win you should **statistically always switch** to the other door.

Mathematically, switching gives you 2/3 (or 66.6…%) chances of winning whereas not switching gives you 1/3 (or 33.3…%) chances of winning.

Let us try to understand why this is true analytically and after that we will look at a formal mathematical proof using probability.

Most people believe that switching or not switching gives 50-50 odds of winning. This would be true if the host opens a door randomly, but that is not the case; the door opened depends on the player's initial choice, so the assumption of independence does not hold. Before the host opens a door there is a 1/3 probability the car is behind each door. If the car is behind door 1 the host can open either door 2 or door 3, so the probability the car is behind door 1 AND the host opens door 3 is $1/3 \times 1/2 = 1/6$. If the car is behind door 2 (and the player has picked door 1) the host **must** open door 3, so the probability the car is behind door 2 AND the host opens door 3 is $1/3 \times 1 = 1/3$. These are the only cases where the host opens door 3, so if the player has picked door 1 and the host opens door 3 the car is twice as likely to be behind door 2. The key is that if the car is behind door 2 the host **must** open door 3, but if the car is behind door 1 the host can open either door.

MATHEMATICAL PROOF USING BAYES' THEOREM

Let the probabilities of the car being behind doors 1, 2 and 3 be P (1), P (2) and P (3) respectively.

$$P(1) = \frac{1}{3} \qquad P(2) = \frac{1}{3} \qquad P(3) = \frac{1}{3}$$

Let the participant choose door 1.

Let P (III / 1), P (III / 2) and P (III / 3) be the probability that the host opens door 3 given that the car was behind door 1, door 2 and door 3 respectively.

$P(III/1) = \frac{1}{2}$ (The host can open door 2 and 3 with equal probability.)

$P(III/2) = 1$ (The host can only open door 3 as door 2 has a car.)

$P(III/3) = 0$ (The host cannot open door 3 as it has a car behind it.)

Now using Bayes' theorem, we try to find P (2 / III) and P (1 / III) that are probabilities of finding cars behind door 2 and door 3 given that door 3 was opened.

$$P(2/III) = \frac{P(III/2)P(2)}{P(III/1)P(1) + P(III/2)P(2) + P(III/3)P(3)}$$

$$= \frac{(1)\left(\frac{1}{3}\right)}{\left(\frac{1}{2}\right)\left(\frac{1}{3}\right) + (1)\left(\frac{1}{3}\right) + (0)\left(\frac{1}{3}\right)} = \frac{2}{3}$$

$$P(1/III) = \frac{P(III/1)P(1)}{P(III/1)P(1) + P(III/2)P(2) + P(III/3)P(3)}$$

$$= \frac{\left(\frac{1}{2}\right)\left(\frac{1}{3}\right)}{\left(\frac{1}{2}\right)\left(\frac{1}{3}\right) + (1)\left(\frac{1}{3}\right) + (0)\left(\frac{1}{3}\right)} = \frac{1}{3}$$

As, we can clearly see P (2 / III) comes out to be twice of P (1 / III), hence we can clearly see that swapping is more beneficial than sticking to the initial choice.

SIMULATION OF "MONTY HALL PROBLEM" IN PYTHON

The program written in python language as shown below simulates the "Monty Hall Problem" for any given number of test cases(N). We can easily change the value of N, the more the number of test cases more accurate is the result observed. We have created the simulation to work by randomly assigning two goats and one car to the doors. Then the program randomly chooses a door and after that the it reveals a goat behind one of the doors. Our program always swaps when given the choice and keeps a track of when it won the car and when it lost and displays the result. The result also shows the configuration of the doors for analysis so we can read and understand each case. A number of comments in the program explain the significance of each line in the program. This python file for this code can be found at: Monty Hall-----

https://github.com/pragati023/Monty_Hall_Birthday_Paradox/blob/b6e21036add244656ada30c62358d7bd411b380d/Monty_Hall_Problem.py

```python
import random
notswapping = 0     # Tracks number of cases where not swapping wins
swapping = 0    # Tracks number of cases where swapping wins
N = 100     # Changes number of cases(increase to increase accuracy)
for i in range(N):
    doors = ["GOAT", "GOAT", "GOAT"]    # List of items behind doors 1, 2, 3
    car_allotment = random.randrange(3)     # A car is allotted to one of the doors randomly
    doors[car_allotment] = "CAR"
    initial_choice = random.randrange(3)    # A door is chosen by the player randomly
    print(f"Case {i+1}:")
    print(f"Player has chosen door {initial_choice+1}")
    host_choices = []   # A list of choices for the host to reveal as part of game
    for j in range(3):
        if (doors[j] != "CAR") and (j != initial_choice):
            host_choices.append(j)
    door_reveal = random.choice(host_choices)   # Host randomly reveals a door which doesn't have a car
    print(f"Host has revealed a goat behind {door_reveal+1}")
    print("Our player has chosen to swap")  # As an experiment to show that swapping is beneficial, our player swaps
    second_choice = -1  # A new choice is made by the user where he/she swaps
    for k in range(3):
        if (k != initial_choice) and (k != door_reveal):
            second_choice = k
    if doors[second_choice] == "CAR":   # If our second_choice has a car
        swapping += 1   # The player won by swapping and it is recorded
        print("Player won by swapping")
    else:       # Otherwise if the car was chosen initially by the player
        notswapping += 1    # The player lost by not swapping and it is recorded
        print("Player lost by not swapping")
    print(doors)    # Prints the items behind each door
    print("\n")
print("Swapping = ", swapping/N)    # Probability that he wins by swapping
print("Not swapping = ", notswapping/N)     # Probability that he wins by not swapping
```

RESULTS OF THE SIMULATION

The results are exported as a pdf file consisting of the 1000 cases for which the experiment was run. Each case shows the whole experiment which can be easily read by the user. Let us analyse the final result:

```
Swapping =  0.662
Not swapping =  0.338
```

The final lines of the file show the probability of winning by swapping and not swapping. As we can see that the result is very close to the expected value of 0.667 and 0.333 as expected. Increasing the number of cases (from 1000) will lead to even better results.

Let us also analyse two of the random cases that were run:

```
Case 944:
Player has chosen door 3
Host has revealed a goat behind 1
Our player has chosen to swap
Player won by swapping
['GOAT', 'CAR', 'GOAT']
```

```
Case 383:
Player has chosen door 1
Host has revealed a goat behind 3
Our player has chosen to swap
Player lost by not swapping
['CAR', 'GOAT', 'GOAT']
```

As we can see in case 944, the configuration was that door 1, door 2 and door 3 had a goat, a car and a goat behind them respectively. The player chose door 3 and after that the host revealed a goat behind door 1. The player chose to switch and hence won the car. In this case swapping won him the car.

As we can see in case 386, the configuration was that door 1, door 2 and door 3 had a car, a goat and a goat behind them respectively. The player chose door 1 and the host revealed a goat behind door 3. The player chose to switch and hence lost the car. In this case swapping caused him to lose the car.

Like these two cases, 1000 cases were run which resulted in the final result discussed above.

The results seem in line with the expected values and hence the simulation is a successful one.

INTRODUCTION TO "BIRTHDAY PARADOX"

Like the previous experiment, the "Birthday Paradox" is also an experiment in probability that seems counterintuitive to many people. The experiment concerns the probability of two people sharing a birthdate (only the date and the month, not the year) in a set of N randomly selected people. The results sometimes may seem absurd to many people but are mathematically true. For example, let us pose a question:

"How many random people should be present in a room for the probability of two people sharing a birthday be greater than 50%?"

The answer seems to be a lot less than people seem to guess.

The answer is a mere **23 people**. That's right, just 23 random people in a room ensures that the probability of two people sharing a birthday is more than 50%.

Another similar type of question that astonishes people is:

"How many random people should be present in a room for the probability of two people sharing a birthday be greater than 99.9%?"

The answer is just **72 people**. Having 72 people in a room almost gives a sure shot chance that two people might be having the same birthday.

This experiment is although based on the assumption that all people are equally likely to be born on any day of the year which doesn't seem to be that absurd of an assumption. For all practical purposes, it can be assumed that birthdays are symmetrically spread out throughout the year.

Let us first discuss the mathematics behind these shocking results and then again carry out a simulation in python language to verify our claims.

MATHEMATICS BEHIND THE "BIRTHDAY PARADOX"

First of all, we should lay down all of the assumptions. It is assumed that a year has 365 days (leap year has not been taken into consideration). It is also assumed that no two people are twins or somehow else related. Also, no seasonal and weekday variation is taken into consideration. It is just assumed that each person present in the room is equally likely to have a birthday on any day of the year. So, let us carry out the calculations for the 50% mark.

Let N be the number of people of people in the room.

When the above assumptions are made, the probability of two random people sharing a birthday is equal to 1/365. Therefore, probability of two people not sharing a birthday is equal to 1 – (1/365) = 364/365. The number of unique pairs which can be formed with N people is ${}^{N}C_{2,}$ that is (N*(N-1))/2. Now the probability of all of these (N*(N-1))/2 pairs having different birthdays is simply

$$\left(\frac{364}{365}\right)^{\frac{N*(N-1)}{2}}$$

But we want that probability of them having same birthday be 50% or 0.5.

$$1 - \left(\frac{364}{365}\right)^{\frac{N*(N-1)}{2}} = 0.5$$

$$\left(\frac{364}{365}\right)^{\frac{N*(N-1)}{2}} = 0.5$$

Taking $\log_{10}$ on both sides,

$$\frac{N*(N-1)}{2} * \log_{10}\left(\frac{364}{365}\right) = \log_{10} 0.5$$

$$\frac{N*(N-1)}{2} \approx 253$$

Which gives, N = 23.

Similarly, for 99.9% mark, we can write the equation as,

$$1 - \left(\frac{364}{365}\right)^{\frac{N*(N-1)}{2}} = 0.999$$

$$\left(\frac{364}{365}\right)^{\frac{N*(N-1)}{2}} = 0.001$$

Taking $\log_{10}$ on both sides.

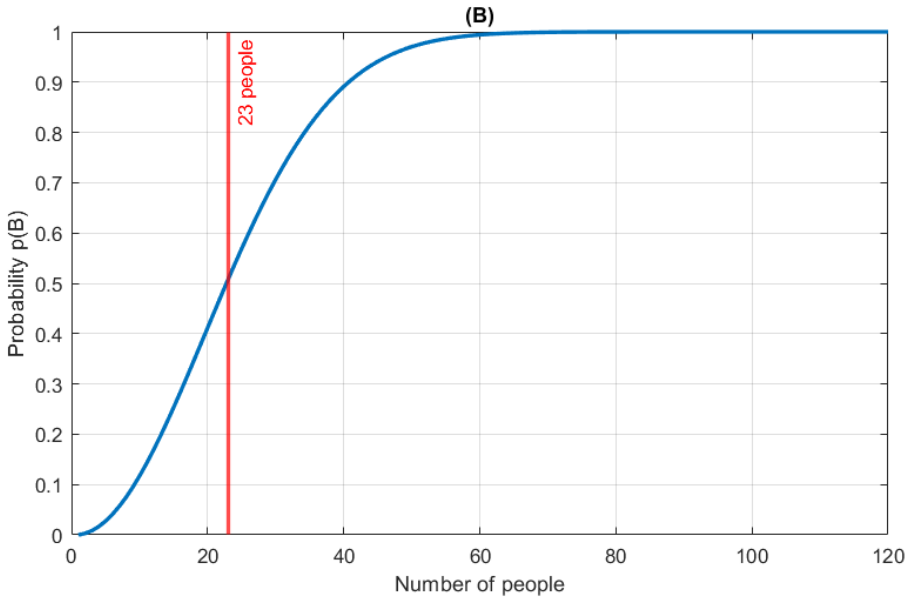
$$\frac{N*(N-1)}{2} * \log_{10}\left(\frac{364}{365}\right) = \log_{10} 0.001$$

$$\frac{N*(N-1)}{2} \approx 2556$$

Which gives, N = 72.

However as one may expect, for the probability of two people sharing a birthday to be one the number of people in the room must be greater than 365. Only that ensures that two people must share a birthday in the room.

Given below is a graph that demonstrates the probability of finding a pair with respect to the number of people.



SIMULATION OF THE "BIRTHDAY PARADOX" IN PYTHON

The simulation for this experiment is pretty simple. The program creates a list of "num" number of people and assigns them a number between 1 and 365 (both inclusive) after that the program tries to find a pair of same dates and if the pair is found or not found the observation made. The program runs "N" of times. Again, increasing N, increases the accuracy of the result. The end result is shown as a statement which shows the probability of two people sharing the same birthday. The python file for this code can be found at:

Birthady Paradox----

https://github.com/pragati023/Monty_Hall_Birthday_Paradox/blob/b6e21036add244656ada30c62358d7bd411b380d/Birthday_Paradox.py

```python
import random
total = 0   # Tracks total cases
pair_exist = 0  # Tracks how many cases have pairs
N = 10000    # Changes number of cases(increase to increase accuracy)
num = 23    # Number of people whose birthdays are simulated
for j in range(N):
    birthdays = []  # List to store list of "num" birthdays
    temp = []   # Temp list - checks presence of a pair
    pair = -1   # Checks status of finding a pair (equal to the day where pair was found)
    check = False   # Checks if a pair is found in the particular case
    for i in range(num):
        birthdays.append(random.choice(range(1, 366)))  # Randomly assigns a day of the year( b/w 1 to 365) to the list
    print(f"Case {j+1}:")   # Prints the case number
    print(birthdays)    # Prints the list of birthdays
    for i in birthdays:     # Checks if a pair exists
        if i not in temp:
            temp.append(i)
        else:
            pair = i    # Stores the day when pair is found
            break
    if pair != -1:
        pair_exist += 1     # Increase the count of cases where pair exist by 1
        check = True
    if check:   # This statement executes when pair is found
        print(f"Pair found at {pair} day.")
    else:   # This statement executes when pair is not found
        print("No pair found.")
    total += 1# Increases the count of total cases by 1
    print("")
print(f"Probability of same birthday = {pair_exist/total}")      # Prints the probability of same birthday
```

RESULTS OF THE SIMULATION

The simulation is run twice, once for “num” = 23 and once for “num” = 72. The pdf files consisting of 5000 cases for “num” = 23 and 2500 cases for “num” = 72 for accuracy are exported as the result. The final results and two random cases from each are discussed below.

```
Probability of same birthday = 0.508
```

```
Probability of same birthday = 0.9992
```

The left snippet is from “num” = 23 case and the right snippet is from “num” = 72 case. Both the results obtained are very close to the expected values of 50% and 99.9% respectively.

```
Case 3716:
[311, 319, 178, 234, 341, 27, 84, 271, 28, 153, 354, 234, 9, 216, 142, 224, 228, 8, 212, 262, 201,
92, 216]
Pair found at 234 day.
```

```
Case 2361:
[218, 65, 176, 3, 131, 23, 6, 122, 155, 42, 280, 91, 300, 275, 72, 22, 33, 363, 303, 228, 53, 10,
144]
No pair found.
```

The above two are snippets from “num” = 23 case where you can see that in case 3716 two people has the same birthday on day 234. Whereas, in case 2361 no two people share the same birthday.

```
Case 581:
[227, 343, 356, 337, 299, 187, 121, 188, 345, 106, 2, 341, 344, 110, 327, 87, 56, 238, 282, 19, 125
, 309, 334, 68, 178, 96, 148, 70, 40, 7, 193, 361, 269, 204, 3, 321, 291, 202, 365, 165, 14, 89, 93
, 65, 182, 138, 217, 61, 36, 313, 296, 306, 66, 173, 218, 340, 258, 293, 62, 264, 324, 177, 216, 116
, 212, 9, 247, 8, 153, 214, 99, 97]
No pair found.
```

```
Case 1495:
[228, 30, 149, 354, 130, 103, 192, 131, 217, 15, 332, 138, 259, 3, 102, 31, 291, 83, 266, 79, 16,
119, 44, 83, 258, 239, 344, 222, 289, 325, 185, 167, 95, 90, 85, 345, 336, 331, 95, 15, 111, 101,
320, 118, 173, 309, 124, 42, 180, 358, 256, 130, 231, 288, 28, 130, 175, 254, 333, 51, 242, 225, 86
, 342, 180, 239, 314, 54, 190, 231, 312, 183]
Pair found at 83 day.
```

The above two are snippets from “num” = 72 case where you can see that in case 581 no two people share their birthdays whereas case 1495 shows the case where two people share their birthdays on day 83. The case 581 is a very rare case as the probability of two people not sharing birthdays is close to 0.001 and this occurs only twice in the 2500 cases! The results are in line with the expected values and the simulation is a successful one.

CONCLUSION AND DISCUSSION

We would like to conclude this report by discussing why the above problems though pretty simple mathematically, seem very absurd and counterintuitive to the human mind.

For example, in the case of Monty Hall Problem, the deep source of confusion is that the **human mind does not understand** that the host **opening the door to reveal a goat does provide additional information** which was previously unknown. People think that they already knew two of the doors had a goat behind them and the host knows where the goat is, so the host revealing a goat does not change the probability of winning by switching. As we have definitely seen mathematically as well as by simulation that switching is always the better option. However, due to sometimes the **brain's ability to think that the choice they already made is best**, they **hesitate to switch** which is better mathematically. Countless surveys show that people would rather stick with their own choice even if they think that switching or not switching is equally likely (which is definitely not the case).

Now, the confusion behind the Birthday Paradox is a completely different one. The main problem why people very wrongly estimate the number of the birthday paradox is that the people really **underestimate the decrement of probability of not finding a pair** which we have seen **goes down exponentially with increasing number of people** in a room. The human brain thinks of probabilities decreasing as "linear" instead of the real deal which is "exponential. Also, humans are a bit selfish and only think about the people their birthday is being compared to and not all the other people who are comparing their birthdays with the other people. All this causes them to misjudge the actual number.

All in all, these two experiments are very simple experiments in probability that tend to confuse the common man as well as mathematicians. We might conclude from these experiments that our intuition many times may not be correct mathematically.

REFERENCES AND BIBLIOGRAPHY

- Monty Hall Problem:
    - https://en.wikipedia.org/wiki/Monty_Hall_problem
    - https://betterexplained.com/articles/understanding-the-monty-hall-problem/
    - https://www.youtube.com/watch?v=4Lb-6rxZxx0
- Birthday Paradox:
    - https://en.wikipedia.org/wiki/Birthday_problem
    - https://betterexplained.com/articles/understanding-the-birthday-paradox/
    - https://www3.nd.edu/~jstiver/Exec_Micro/Monty%20Hall.pdf
    - https://www.statisticshowto.com/probability-and-statistics/monty-hall-problem/#:~:text=As%20Monty%20has%20opened%20door,that%20is%20why%20you%20switch.
- Python and Simulation:
    - https://www.geeksforgeeks.org/birthday-paradox/
    - https://www.python.org/
- Link to python codes and report:

    https://github.com/pragati023/Monty_Hall_Birthday_Paradox