

## FINDING THE BEST STRATEGY IN 2-PLAYER GAMES THROUGH ITERATION

Aashman Trivedi\*<sup>1</sup>, Abhinav Palanivel\*<sup>2</sup>, Aarnav Thirumal\*<sup>3</sup>

\*<sup>1,2,3</sup>Student, IBDP, Canadian International School, Bangalore, India.

DOI : <https://www.doi.org/10.56726/IRJMETS44020>

### ABSTRACT

Various games can have multiple outcomes, based on the decisions of the various stakeholders or players. In this paper, we created a program (in Python) that finds the type of dominant strategy in 2-player games, using the potential payoffs or utility returns for each of the possible moves a player plays. Specifically, the code was ideally made for a game where each of the players has 2 moves thus 4 total possible outcomes for each player with their own payoffs. We were able to process the inputted data for each of the 2 players through the use of a 2:2 matrix. By comparing payoffs or specific values in the matrix using logical steps and conditions, we were able to determine what move would be most beneficial and effective for each player, along with the type of dominance that move has.

**Keywords:** Strategy, Utility, Payoff, Nash Equilibrium, Game Theory.

### I. INTRODUCTION

When coding for various different strategies inside of a game, it is often useful to first gain a qualitative understanding of these different strategies and what a Nash equilibrium is. This is so that we can then apply this theory in order to quantitatively approach the strategies for given games, and come up with an effective methodology that is logical and one that can be programmed.

**Nash Equilibrium:** The Nash Equilibrium is the point or position at which no player would want to deviate. It is also where the payoffs are the highest for each of the players, which is why the players don't want to deviate from this particular position.

**Strongly Dominant Strategy:** This is a strategy in which no matter what the other players play, the payoff will always be higher by playing this particular move.

**Weakly Dominant Strategy:** This is a strategy in which the payoff will be higher than or equal to (never less than) playing another strategy, and this depends on what the other player plays.

**Very Weakly Dominant Strategy:** This is a strategy in which irrespective of what move you or the other players pick, the payoffs will be the same.

**Pure Strategy:** This is where there is no dominant strategy as there is no particular move where irrespective of what the other players play, you will get a greater or equal payoff. Therefore, getting a greater payoff after playing any move is purely based on what the other player plays.

The goal of the program is not to be as efficient as possible, but rather to help users find the best strategies in various games, and further understand why certain moves are better to play compared to others using simple logic that can be understood via the code and this paper, which is also thoroughly commented for ease of understanding.

### II. METHODOLOGY

Now that a qualitative definition of the various strategies we will be programming for is understood in theory, here is how they play out in real life. Using the following conditions which make up the methodology that is used in our program, we can logically figure out which strategy is best for each player to play.

Real example are included with payoffs represented inside of a 2:2 matrix, where there are Players 1 and 2. Player 1 has to choose from the options in the first column, while player 2 has to choose from the options in the first row. Each player has 2 labelled options, and their payoff for each possible scenario is listed inside of brackets (player 1, player 2).

**Strictly Dominant Strategy (SDS)**

P1/P2	Home	MG Road
Home	(0, 0)	(0, 1)
MG Road	(1, 0)	(2, 2)

In this matrix each player has 2 options; staying at home or going to MG Road.

If P2 stays home, P1 gets a payoff of 0 and 1 respectively if they go home or to MG road, where  $1 > 0$  so going to MG road is better. If P2 goes to MG road however, P1 gets a payoff of 0 and 2 respectively where  $2 > 0$  and thus going to MG road gives a greater payoff again.

The exact same logic can be applied for P2. If P1 stays home, P2 gets a payoff of 0 and 1 respectively if they go home or to MG road, where  $1 > 0$  so going to MG road is better. If P1 goes to MG road however, P2 gets a payoff of 0 and 2 respectively where  $2 > 0$  and thus going to MG road gives a greater payoff again.

The Strictly Dominant Strategy is going to MG Road for both players as it has a greater payoff no matter what the opponent chooses. This box (right corner) is also where the Nash Equilibrium lies as the payoffs are highest for both players here and no player would want to deviate from this position.

**Weakly Dominant Strategy (WDS)**

P1/P2	Left	Right
Left	(2, 2)	(3, 1)
Right	(1, 3)	(3, 3)

In this matrix each player has 2 options; going left or right.

If P2 goes Left, P1 gets a payoff of 2 going left and 1 going right, and since  $2 > 1$ , he should go left in this case. However, if P2 goes right, P1 gets a payoff of 3 no matter where he goes ( $3 = 3$ ).

We can use the exact same logic for P2. If P1 goes Left, P2 gets a payoff of 2 going left and 1 going right, and since  $2 > 1$ , he should go left in this case. However, if P1 goes right, P2 gets a payoff of 3 no matter where he goes ( $3 = 3$ ).

Since  $2 > 1$  and  $3 = 3$ , going left for Player 1 and Player 2 is the Weakly Dominant Strategy Equilibrium in this case, as irrespective of what the other player plays, a greater or equal payoff is obtained by going left for both players.

**Very Weakly Dominant Strategy (VWDS)**

P1/P2	North	South
North	(-2, -2)	(-5, -2)
South	(-2, -10)	(-5, -10)

In this matrix each player has 2 options; going North or South.

The other player going North gives P1 and P2 a payoff of -2 irrespective of what the other player plays, while the other player going south gives P1 and P2 a payoff of -5 and -10 respectively, irrespective of what the other player plays. Thus, each players' payoff will be the same no matter what move they make.

All Strategies are very weakly dominant as all the payoffs are equal in each case, regardless of what the other player moves.

**Pure Strategy (PS)**

P1/P2	Stag	Hare
Stag	(3, 3)	(0, 2)
Hare	(2, 0)	(1, 1)

In this matrix each player has 2 options; hunting a stag or a hare.

If P2 hunts a stag, P1 gets a payoff of 3 and 2 respectively if they hunt a stag or hare, where  $3 > 2$  and thus hunting a Stag is better. If P2 hunts a hare however, P1 gets a payoff of 0 and 1 respectively, where  $1 > 0$  and thus hunting the hare is the better option in this case.

The exact same logic can be applied for P2. If P1 hunts a stag, P2 gets a payoff of 3 and 2 respectively if they hunt a stag or hare, where  $3 > 2$  and thus hunting a Stag is better. If P1 hunts a hare however, P2 gets a payoff of 0 and 1 respectively, where  $1 > 0$  and thus hunting the hare is the better option in this case.

This is a Pure Strategy for both players because there is no dominant strategy here which ensures a greater or equal payoff by playing a particular move irrespective of what the other person plays.

**III. PROGRAMMING**

The program itself is largely based on the methodology explained above. We also fully commented the code for better structure and readability. It is noteworthy that exact code for player 1 was replicated for player 2 (renamed variables were used with the same function along with slight adjustments in the logic that are mentioned in our methodology), so we won't go over that; however the full version of the code is attached here for use and viewing.

Also, like stated in the introduction, the focus of the code was not on maximizing efficiency. One major way we could have shortened the code would've been through adding a certain loop along with a few short lines of code to repeat and slightly adjust the methodology which is slightly different for player 2, rather than fully copying the code twice and then making those adjustments. However, the code does still work and we did focus on ease of understanding.

```

1 #Define Matrix
2 #Player 1
3 N = int(input('Enter the number of rows/columns for player 1: '))
4 matrix = []
5 print('Enter the payoffs columnwise:')
6 #User input
7 for i in range(N):
8     a = []
9     for j in range(N):
10        a.append(float(input()))
11    matrix.append(a)

```

**Figure 1:** The first part of the code where the matrix and user input is defined

```

13 #VWDSE
14
15 count = 0
16 for i in range(N):
17     if matrix[i].count(matrix[i][0]) == N:
18         count += 1

```

**Figure 2:** The code that allows a very weakly dominant strategy to be identified.

```

20 #SDSE and WDSE
21
22 countfour = 0
23 Arr = [0] * N
24 for i in range(N):
25     Arr[i] = max(matrix[i])
26     if matrix[i].count(Arr[i]) == N:
27         countfour += 1
    
```

**Figure 3:** The code that allows a strongly dominant or weakly dominant strategy to be identified. These are clustered into one iterative loop for convenience and efficiency.

```

29 #PNSE
30
31 countthree = 0
32 Arr2 = [0] * N
33 for i in range(N):
34     Arr2[i] = max(matrix[i])
35     b = matrix[0].index(Arr2[0])
36     if b != matrix[i].index(Arr2[i]):
37         countthree += 1
    
```

**Figure 4:** The code that allows a pure strategy Nash equilibrium to be identified.

```

39 #Count and Print
40 if count == N:
41     print('All strategies are very weakly dominant for player 1')
42 elif countfour >= 1:
43     print('Strategies at the row with payoffs:', Arr,
44           'are weakly dominant for player 1')
45 elif countthree == N - 1:
46     print('This game has a pure Nash Equilibrium')
47 elif countfour == 0:
48     print('Strategies at the row with payoffs:', Arr,
49           'are strongly dominant for player 1')
    
```

**Figure 5:** The code that defines the output.

#### IV. RESULTS AND ANALYSIS

We will be inputting the examples shown above into our program, to ensure that it outputs the correct result and matches the methodology explained and used inside the program. I have copied the matrix of payoffs for each move below for convenience.

##### Strongly Dominant Strategy (SDS)

P1/P2	Home	MG Road
Home	(0,0)	(0,1)
MG Road	(1,0)	(2,2)

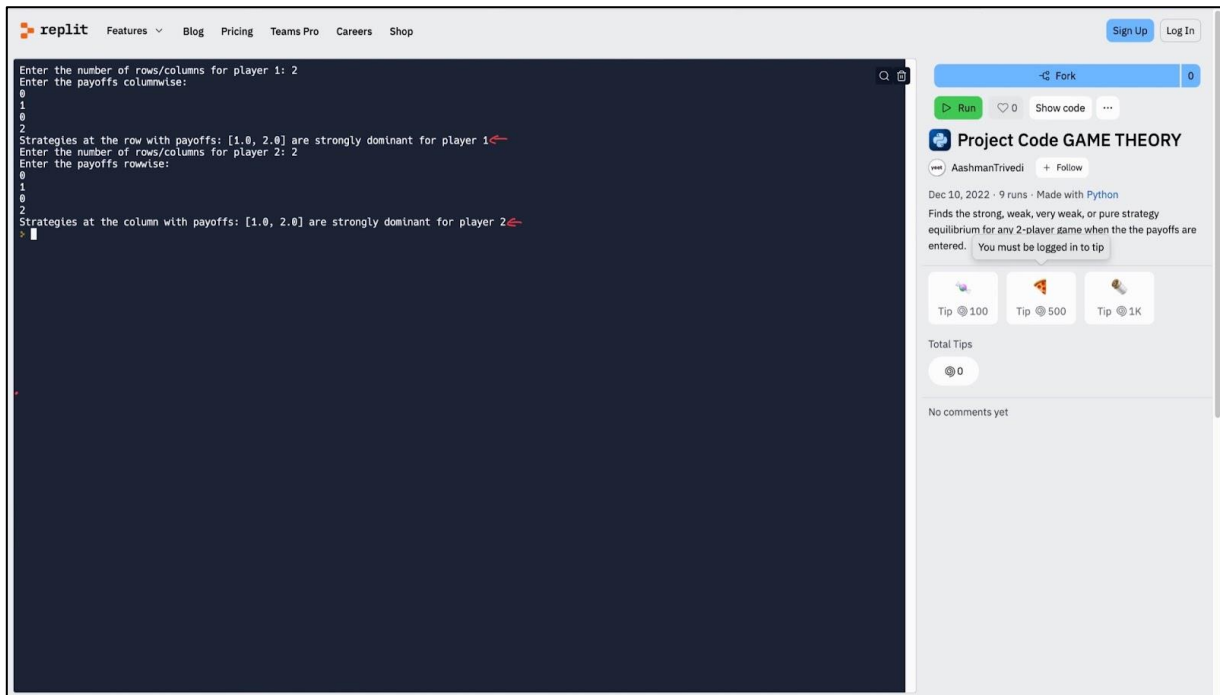


Figure 6: The strongly dominant strategy correctly identified and outputted using the code (for the matrix above).

**Weakly Dominant Strategy (WDS)**

P1/P2	Left	Right
Left	(2, 2)	(3, 1)
Right	(1, 3)	(3, 3)

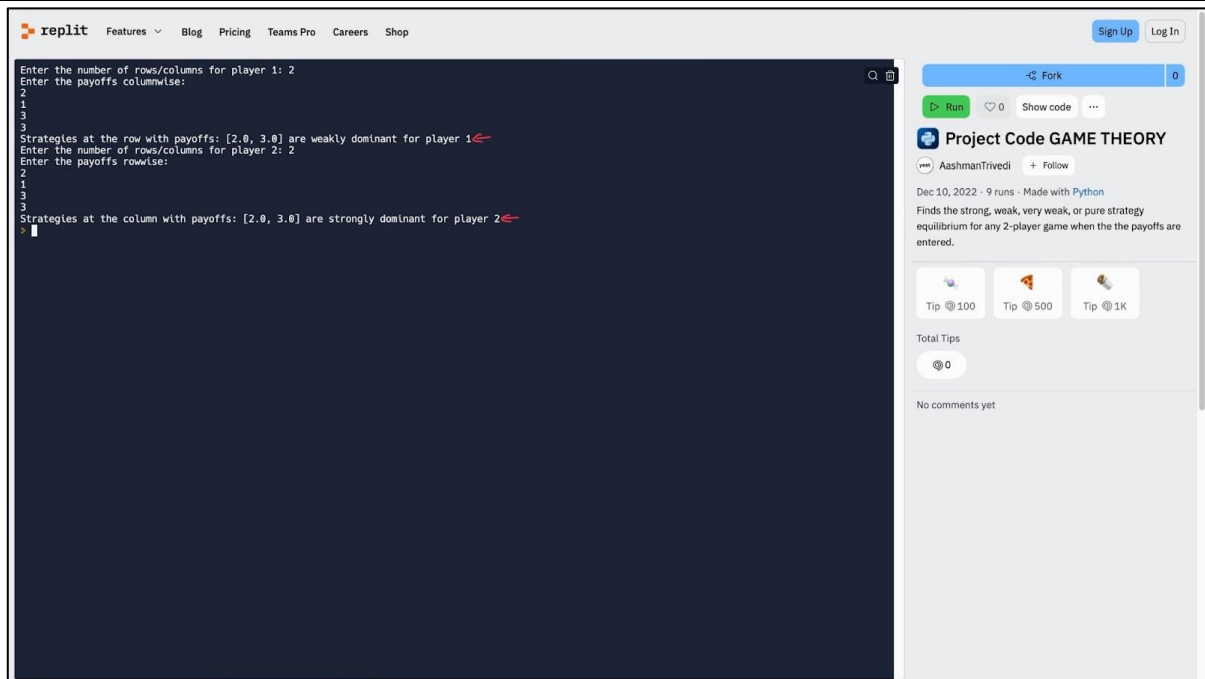
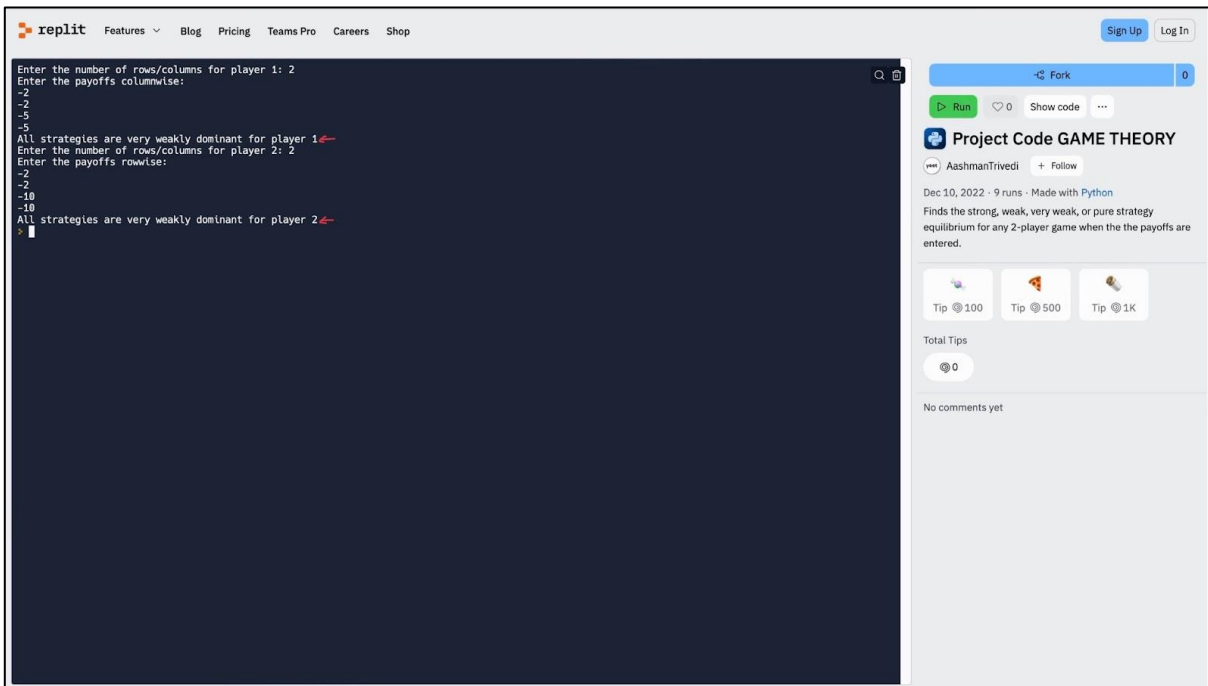


Figure 7: The weakly dominant strategy correctly identified and outputted using the code (for the matrix above).

**Very Weakly Dominant Strategy (VWDS)**

P1/P2	North	South
North	(-2, -2)	(-5, -2)
South	(-2, -10)	(-5, -10)



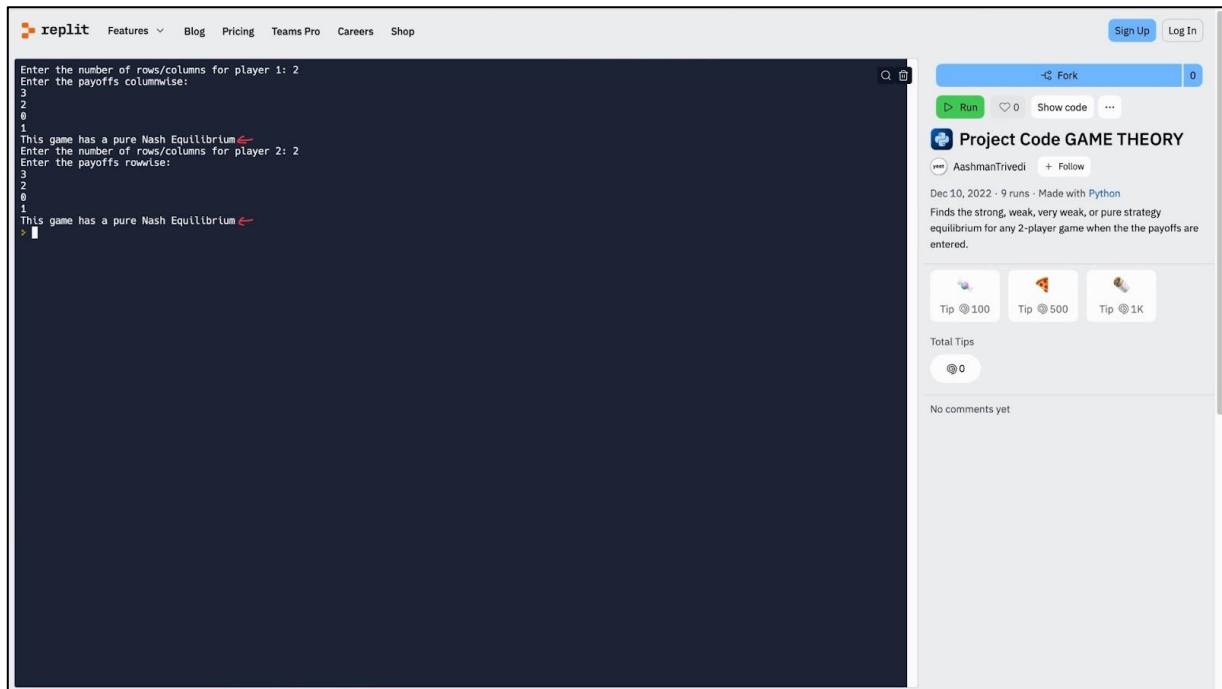
```

replit Features Blog Pricing Teams Pro Careers Shop Sign Up Log In
Enter the number of rows/columns for player 1: 2
Enter the payoffs columnwise:
-2
-2
-5
-5
All strategies are very weakly dominant for player 1
Enter the number of rows/columns for player 2: 2
Enter the payoffs rowwise:
-2
-2
-10
-10
All strategies are very weakly dominant for player 2
>
  
```

**Figure 8:** The very weakly dominant strategies were correctly identified and outputted using the code (for the matrix above).

**Pure Strategy (PS)**

P1/P2	Stag	Hare
Stag	(3, 3)	(0, 2)
Hare	(2, 0)	(1, 1)

The screenshot shows a Replit terminal window with a dark background. The terminal text is as follows:

```
replit Features Blog Pricing Teams Pro Careers Shop Sign Up Log In
Enter the number of rows/columns for player 1: 2
Enter the payoffs columnwise:
3
2
0
1
This game has a pure Nash Equilibrium ←
Enter the number of rows/columns for player 2: 2
Enter the payoffs rowwise:
3
2
0
1
This game has a pure Nash Equilibrium ←
>
```

On the right side of the terminal, there is a sidebar for a project titled "Project Code GAME THEORY" by AashmanTrivedi. It shows the code was run on Dec 10, 2022, and includes a description: "Finds the strong, weak, very weak, or pure strategy equilibrium for any 2-player game when the the payoffs are entered." There are also options to tip (100, 500, 1K) and a "Total Tips" section showing 0.

**Figure 9:** The pure strategy equilibrium was correctly identified and outputted using the code (for the matrix above).

## V. CONCLUSION

It can be concluded that after inputting a 2:2 matrix, and then using the logic explained in the methodology above, you can find the move which would be best for any of the players and how dominant that strategy is.

## ACKNOWLEDGEMENTS

I would like to express my sincere gratitude towards Professor Y.Narahari for inspiring me to write this research paper. He extensively taught me some of the ideas that are utilized in this paper and helped me further refine this paper through his valuable suggestions.

## VI. REFERENCES

- [1] Narahari, Y. [2014]. Game Theory and Mechanism Design [Vol. 4].
- [2] Allen B (1997) Implementation theory with incomplete information. Cooperation: game-theoretic approaches, pp 115–126.
- [3] Ausubel LM, Cramton P, Deneckere RJ (2002) Bargaining with incomplete information. Handbook of game theory with economic applications, vol 3, pp 1897–1945.